Algoritmi

Un approccio storico-didattico ad un leitmotiv della Matematica

di
Aldo Scimone

$$\frac{4}{\pi} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{7^2}{2 + \cfrac{9^2}{2 + \ddots}}}}}$$

Considerazioni generali

The word "algorithm" itself is quite interesting; at first glance it may look as though someone intended to write "logarithm" but jumbled up the first four letters.

Donald E. Knuth, The Art of Computer Programming, *v. I, p.1.*

Non appena una cosa viene dimostrata impossibile a compiersi, una quantità di gente si mette al lavoro per cercare di realizzarla comunque. Questa sembra una componente essenziale del comportamento umano.

Donald E. Knuth, Algoritmi, "Le Scienze" n. 108, agosto 1977, p. 14.

Algoritmo:

La forma più antica di questo termine si trova nel latino medievale, in cui, con la parola *algorithmus* o *algorismus*, si designava ogni procedimento per eseguire operazioni aritmetiche facendo uso delle cifre arabe che erano state introdotte in occidente con l'opera *Liber Abaci*, del 1202, di Leonardo Pisano, il grande matematico italiano noto soprattutto con il soprannome di *Fibonacci*.

Ritratto di Fibonacci

Nel Medioevo, infatti, veniva fatta una distinzione tra i cosiddetti *abacisti*, che calcolavano con l'abaco, e gli *algoristi*, che calcolavano usando appunto le *nuove* cifre arabe. Nel Rinascimento si riteneva che l'origine della parola fosse dovuta alla combinazione delle parole *algiros* (penoso) e *arithmos* (numero); mentre altri ritenevano che derivasse da "Re Algor di Castiglia". Finalmente, gli storici della matematica hanno stabilito che il termine medievale *algorismus* deriva

dal nome del matematico persiano *Abu 'Abd Allah Muhammad ibn Musa al-Khwarizmi* (c. 825) - letteralmente, "*Padre di Abdullah, Mohammed, figlio di Moses, nativo di Khwarizm.*" Khwarizm era una regione dell'Asia centrale, localizzata nel bacino del fiume Amu, a sud del Mar d'Aral che un tempo era noto come Lago Khwarizm. *al-Khwarizmi* scrisse l'opera *Kitab Al-jabr wa'l Muqabala* (L'arte di numerare ed ordinare le parti in un tutto) da cui, per inciso, deriva il nome *Algebra.*

La traduzione di questo trattato in latino fu fatta molti anni dopo e cominciava con le parole *Dixit Algorithmi*, storpiando il soprannome del suo autore; da qui ebbe origine il termine *algoritmo.*

Tutta la storia della Matematica è costellata di *algoritmi,* fin dalle testimonianze più remote. In realtà le radici dell'algoritmica sono molto antiche. Infatti, anche se l'assetto teorico di questa disciplina è stato raggiunto solo nella prima metà del novecento, e le tecniche di progetto e di analisi degli algoritmi hanno avuto enormi progressi con la diffusione dei calcolatori elettronici, i primi esempi di algoritmi risalgono alle origini della storia dell'uomo, e ne sono rimasti esempi nei documenti matematici più antichi.

Fino ad alcuni decenni fa la parola «algoritmo» era sconosciuta alla maggior parte delle persone colte, ma lo sviluppo rapidissimo avuto dalla scienza dei calcolatori, nell'ultimo ventennio del ventesimo secolo, ha modificato la situazione, ed attualmente la parola è diventata indispensabile, e le pubblicazioni riguardanti gli algoritmi hanno raggiunto un numero considerevole. Esistono diversi termini che, almeno in parte, esprimono il concetto in questione, come: procedura, prescrizione, routine, processo, metodo; e al pari di queste cose, in prima battuta, si può definire un algoritmo come *un insieme di regole o direttive atte a fornire una risposta specifica a una specifica richiesta.* Possiamo aggiungere anche subito che caratteristica distintiva degli algoritmi è *la totale eliminazione delle ambiguità*: le regole devono essere sufficientemente semplici e ben definite da poter essere eseguite da una macchina. Un'altra caratteristica fondamentale degli algoritmi è che devono sempre *avere termine* dopo *un numero finito di passi.*

Anche se tradizionalmente gli algoritmi sono stati applicati unicamente a problemi numerici, tuttavia l'esperienza con i calcolatori ha mostrato che i dati elaborati dai cosiddetti *programmi* (esposizioni degli algoritmi in un linguaggio accuratamente definito) possono rappresentare virtualmente qualsiasi cosa. Ciò ha fatto sì che gli studiosi abbiano cominciato ad interessarsi maggiormente delle strutture con cui si possono rappresentare le informazioni e all'aspetto ramificato o decisionale degli algoritmi, che permette di seguire differenti sequenze di operazioni dipendenti dallo stato delle cose in un determinato istante.

Lo studio sistematico degli algoritmi, la loro definizione matematica, l'analisi delle operazioni da eseguire, la precisazione delle classi di problemi risolubili mediante algoritmi sono stati oggetto di ricerca nel secolo scorso. A tale riguardo, studi importanti vennero eseguiti da Andrei Andreievic Markov (1856-1922); all'inizio degli anni trenta venne inoltre sviluppata la teoria delle funzioni ricorsive da parte di molti matematici, tra i quali spicca il grande Kurt Gödel (1906-1978).

Kurt Gödel

Alan Turing

Anche la teoria delle macchine 'computazionali', che è strettamente collegata allo studio degli algoritmi, venne sviluppata a partire dagli anni trenta da Alan Turing (1912-1954), Emil Leon Post (1897-1954) e Alonzo Church.
Al concetto di algoritmo spetta, quindi, un ruolo centrale nell'*informatica,* e l'algoritmica è appunto quella parte dell'informatica rivolta alla *definizione*, *costruzione* e *analisi* di comportamento degli algoritmi.
Anche noi, fin da quando cominciamo la nostra formazione matematica di base alla scuola elementare, veniamo subito a contatto con diversi algoritmi; come, per esempio, nel momento in cui apprendiamo le procedure per sommare, sottrarre, moltiplicare e dividere i numeri naturali.

Alcuni esempi di algoritmi

a] Uno degli algoritmi più antichi venne scoperto su alcune tavolette di argilla in Mesopotamia e si riferisce al calcolo della somma delle potenze del numero *2* con esponente variabile da *1* ad *n*. Rappresentandolo nel nostro linguaggio, esso si compone dei passi seguenti:

1. Inizio dell'algoritmo.

2. Somma la potenza di *2* da *1* ad *n*, con *n* numero intero ≥ 1.

3. L'ultimo termine della somma è 2^n.

4. Sottrai *2* da 2^n, troverai quindi $2^n - 2$.

5. Somma $(2^n - 2)$ a 2^n e ottieni la risposta; quindi la somma è $S = 2^n + (2^n - 2)$.

6. Questo è il procedimento.

Algebricamente, questo procedimento della matematica babilonese viene convalidato dalla formula che fornisce la somma delle potenze di *2*, quando l'esponente *k* assume tutti i valori interi positivi da 1 ad *n*:

$$\sum 2^k = 2^n + (2^n - 2)$$

Questa formula rappresenta, infatti la somma di *n* termini della progressione geometrica:

$$2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, \ldots$$

di ragione *2*, per cui la somma vale:

$$S_n = a_1 \cdot \frac{q^n - 1}{q - 1} = 2 \cdot \frac{2^n - 1}{2 - 1} = 2 \cdot 2^n - 2 = 2^n + (2^n - 2).$$

Ciò che appare subito evidente è che qualsiasi persona, pur non conoscendo il concetto di progressione geometrica, può raggiungere il risultato desiderato, seguendo le singole istruzioni del procedimento descritto.

b] Nel famoso *Papiro Rhind*, che si ritiene scritto dallo scriba egizio *Ahmes*, circa nel 1650 a.C., e conservato nel British Museum di Londra, spicca un algoritmo di moltiplicazione tra interi che coincide, sorprendentemente, con il nostro usuale algoritmo di moltiplicazione, se gli operandi fossero rappresentati in base *2*; solo che gli egiziani non conoscevano tale rappresentazione!

Una parte del Papiro Rhind

Tale algoritmo può essere descritto mediante i passi seguenti.

1] Sono dati i numeri interi positivi *a* e *b*. Sia *p* il loro prodotto, inizialmente incognito.
2] Poni $p = 0$ e controlla se *a* è dispari o pari.

3] Finché $a \neq 0$ ripeti la seguente procedura:
3.1 se *a* è dispari, allora fai la somma di *b* e di *p*;
3.2 Dimezza *a* trascurando il resto;
3.3 Raddoppia *b*.

4] Scrivi il risultato finale.

Per esempio se $a = 45$ e $b = 14$, il prodotto $p = 630$ si costruisce come segue:

p :	*0*	*14*	*14*	*70*	*182*	*182*	*630*
a :	*45*	*22*	*11*	*5*	*2*	*1*	*0*
b :	*14*	*28*	*56*	*112*	*224*	*448*	*896*

In questo esempio, se rappresentiamo *45* in base *2* otteniamo:

1 0 1 1 0 1

Scriviamo questo numero in forma polinomiale:

$$2^5 + 2^3 + 2^2 + 2^0$$

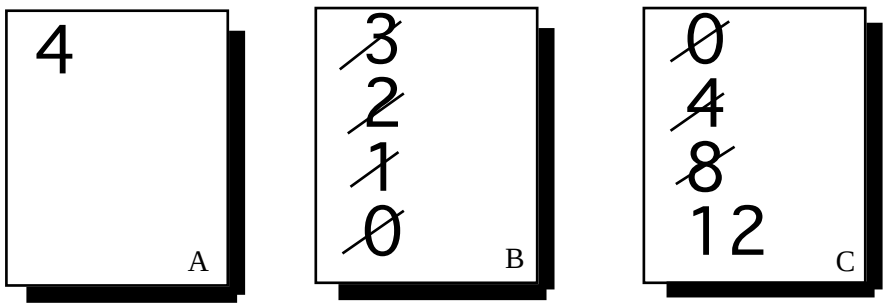
Se eseguiamo ora la moltiplicazione per *14*, si ha:

$$(2^5 + 2^3 + 2^2 + 2^0) \cdot 14 = 448 + 112 + 56 + 14 = 630$$

che è la somma corrispondente al prodotto $45 \cdot 14$.

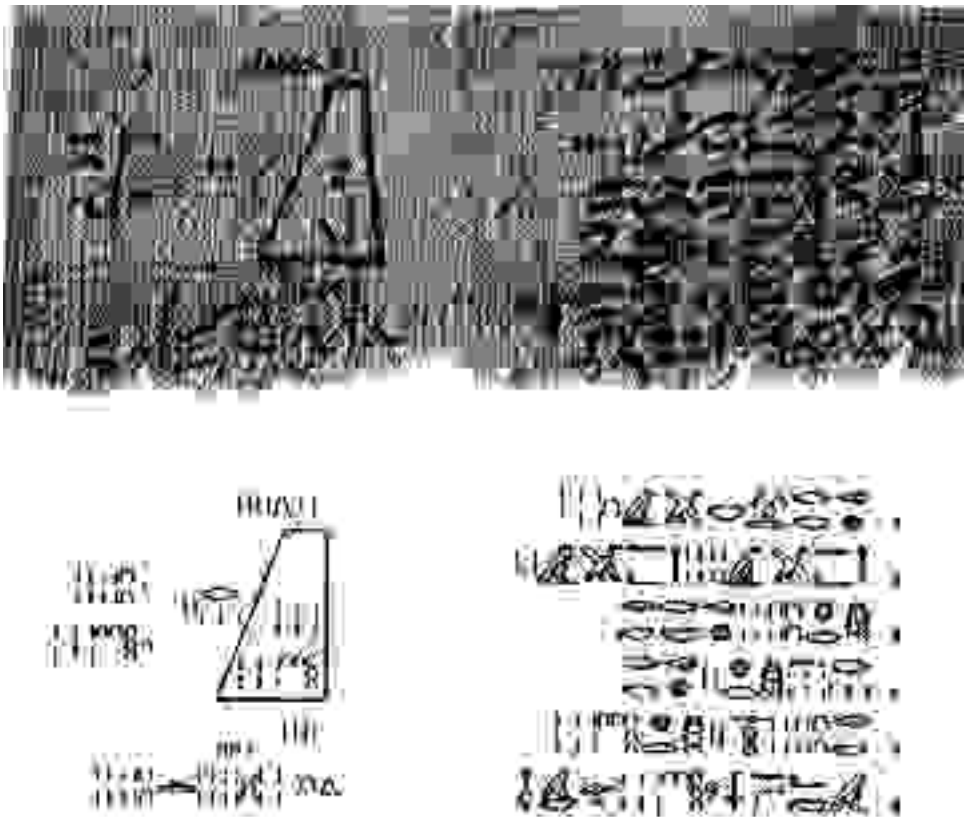
Un procedimento simile potremmo utilizzarlo per fare eseguire la moltiplicazione tra due numeri anche ad un allievo che conosce i simboli *1, 2, 3, ...*, che sa sommare e sottrarre, che possiede fogli di carta, ma non sa che cosa significhi «*moltiplicare*». Potremmo, per esempio, descrivere l'operazione nel modo seguente, pur sapendo che il metodo descritto sarebbe lontano dalla nostra idea di «*moltiplicazione*»:

Inizio
    scrivi il numero *a* sul foglio A
    scrivi il numero *b* sul foglio B
    scrivi il numero *0* sul foglio C
    finché sul foglio B c'è un numero maggiore di *0* ripeti le seguenti istruzioni:
        inizio
            fai la somma del numero scritto sul foglio C col numero scritto sul foglio A
            scrivi il risultato sul foglio C dopo aver cancellato il numero precedente
            sottrai 1 al numero scritto sul foglio B e scrivi il risultato sul foglio B
            dopo aver cancellato il numero precedente
        fine
    leggi il numero scritto sul foglio C
fine.

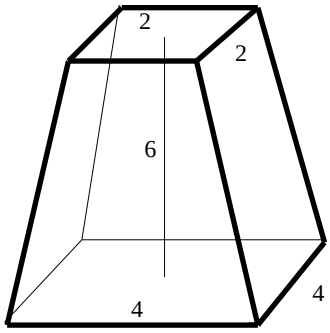


c] Un altro *calcolo guidato* si trova nel *Papiro di Mosca*, risalente al 1700 a.C., in cui sono risolti *25* problemi della stessa natura di quelli del *Papiro Rhind*. Uno dei più interessanti è il seguente, in cui si calcola il volume di un tronco di piramide a base quadrata secondo questo metodo:

Se ti viene detto: c'è una piramide tronca che ha 6 per altezza verticale, 4 per la base e 2 per la cima. Fai il quadrato di questo 4, risultato 16. Raddoppia 4, risultato 8. Fai il quadrato di 2, risultato 4. Addiziona il 16, l'8 e il 4, risultato 28. Poi prendi un terzo di 6, risultato 2. Allora fai due volte 28, risultato 56. Vedrai che il risultato è giusto.



Il problema del tronco di piramide nel Papiro di Mosca



L'algoritmo si compone, quindi, dei seguenti passi:

1. Inizio dell'algoritmo (altezza 6, base 4, cima 2).
2. Fai il quadrato di 4.
3. Scrivi il risultato: 16.
4. Fai il doppio di 4.
5. Scrivi il risultato: 8.
6. Fai il quadrato di 2.
7. Scrivi il risultato: 4.
8. Somma 16 + 8 + 4.
9. Scrivi il risultato: 28.
10. Calcola $\frac{1}{3}$ di 6.

11. Scrivi il risultato: 2.
12. Moltiplica questo 2 per 28.
13. Scrivi il risultato: 56.
14. Fine dell'algoritmo.

Ciò che meraviglia maggiormente in questi esempi è che essi conducono il lettore al risultato giusto, ma senza motivare la successione delle operazioni. Essi appaiono *incredibilmente* come algoritmi il cui esecutore è pensato più come un *automa* che come un essere umano. Il tono è assolutamente apodittico, senza che colui che propone il calcolo senta il bisogno di dimostrare l'efficacia e l'attendibilità del procedimento. Nel caso del tronco della piramide, poi, non viene neanche fornito un metodo generale valido per tutti i tipi di questi solidi. Non si può sperare, in questo tipo di geometria pratica di trovare una formula generale come quella che usiamo noi per trovare il volume di qualsiasi tronco di piramide, conoscendo il lato *a* della base maggiore, il lato *b* della base minore e l'altezza *h*:

$$V = \frac{1}{3} h (a^2 + ab + b^2)$$

L'unica indicazione è quell'affermazione finale: "*vedrai che il risultato è giusto*", che sembra più un voler dire: "*non preoccuparti, è come ti dico io*", senza che il lettore possa chiedere: "*ma perché si deve fare così?*"

d] Consideriamo, ora, un algoritmo per ordinare un insieme di elementi, siano essi numeri o parole.

Per capire come costruire l'algoritmo, supponiamo che i numeri siano tre:

5, 3, 2

Il primo passo consisterà nel confrontare il primo numero con il secondo ed effettuare uno scambio nel caso in cui il primo sia maggiore del secondo. Nel nostro caso otteniamo:

3, 5, 2

Ora confrontiamo il primo numero con il terzo ed effettuiamo lo scambio dopo il confronto, qualora sia necessario. Nel nostro caso otteniamo:

2, 5, 3

Ora confrontiamo il secondo numero con il terzo, ed effettuiamo lo scambio, se necessario. Otteniamo:

2, 3,5.

Abbiamo finito, perché i tre numeri ora sono ordinati.

I passi dell'algoritmo potrebbero essere, allora, i seguenti:

1. Siano *a, b, c* i tre numeri dati;
2. Se $a > b$, allora scambia *a* con *b*, altrimenti non fare nulla;
3. Se $a > c$, allora scambia *a* con *c*, altrimenti non fare nulla;
4. Se $b > c$, allora scambia *b* con *c*, altrimenti non fare nulla;
5. Scrivi l'ultima sequenza dei numeri.

e] Consideriamo ora un esempio di algoritmo di natura diversa dai precedenti, relativo al "*gioco dell'undici*".

Su di un tavolo vengono posti undici oggetti, uguali o diversi. Due giocatori, A e B, devono raccogliere a turno da uno a tre degli oggetti. Perde chi raccoglie l'ultimo oggetto.

Ebbene, se il giocatore *A* deve fare la prima mossa, allora egli potrà essere sicuro di vincere se seguirà le seguenti istruzioni di gioco:

1. *A* raccoglie *2* oggetti.
2. Il gioco passa al giocatore *B.*
3. *B* raccoglie *k* oggetti, con $1 \leq k \leq 3$.
4. Finché sul tavolo ci sono oggetti, *A* raccoglie $4 - k$ oggetti e si torna al passo 2.
5. *A* vince.

Questo algoritmo è stato portato come esempio di quella che dev'essere l'*efficacia* di un algoritmo. In questo caso l'algoritmo risolve una classe di problemi e non un singolo problema, perché i problemi sono tanti quante sono le possibili mosse del giocatore *B.* Infatti, *B* può prendere *1*, *2* o *3* oggetti, ogni volta che viene il suo turno.

Nell'uso odierno, per algoritmo si intende, sinteticamente, qualunque schema o procedimento di calcolo; più precisamente:
Un procedimento di calcolo esplicito e descrivibile con un numero finito di regole che conduce al risultato dopo un numero finito di operazioni, cioé di applicazione delle regole.

Anche se il concetto di algoritmo ha una connotazione precisa solo nella scienza, e in particolare nella matematica, esso, contrariamente all'opinione comune, non è un patrimonio solo dell'ambito scientifico, ma, in un senso più lato, sta anche alla base di molte azioni che di solito vengono dette *abitudinarie*, ma che, a ben analizzarle, possono essere decodificate mediante successioni di atti che il nostro cervello ha ormai immagazzinato in maniera definitiva e che, ripetute nel loro complesso, appaiono *naturali.*
Tali sono, per esempio, le azioni che riguardano:

- l'aprire una porta;

- prendere l'ascensore;

- fare una telefonata;

- mettere in moto l'automobile.

f] Consideriamo, ad esempio, l'azione di *"fare una telefonata"*. Alla base dell'azione c'è la necessità di risolvere un problema: comunicare qualcosa ad un'altra persona che è irraggiungibile fisicamente, e che vive nella nostra stessa città o in un'altra città italiana.
Ecco allora che ci avviciniamo all'apparecchio telefonico, alziamo il ricevitore, componiamo il numero e attendiamo che la persona chiamata ci risponda. Se risponderà, allora avremo risolto il nostro problema; in caso contrario, potremo decidere di riprovare dopo qualche tempo.
Se analizziamo, nelle sue linee essenziali, la successione di azioni che si devono compiere, possiamo elencarle nel modo seguente:

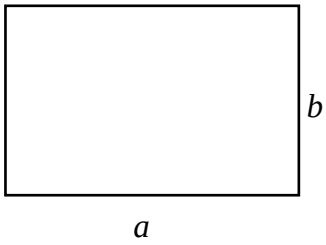
1] mi avvicino all'apparecchio telefonico;
2] sollevo il ricevitore;
3] attendo il segnale acustico;
4] compongo il numero;
5] se il numero chiamato è libero, attendo la risposta, altrimenti riaggancio il ricevitore.

Chiediamoci: a quali criteri soddisfa questa sequenza di azioni? Certamente, essa ***è finita***, cioè, termina in un tempo finito; inoltre, ***non crea contraddizioni***, ovvero, non mi porta a fare

un'azione che sia in contrasto con un'azione fatta precedentemente nella sequenza; infine, ***fa sempre la stessa cosa***, cioé, ogni volta che io ripeterò la successione delle azioni , come risultato finale potrò comunicare con la persona chiamata oppure no. Possiamo, quindi, concludere che questo algoritmo è *buono*, ovvero *risolve efficacemente* il nostro problema.

g] Consideriamo un altro problema, questa volta di natura matematica:

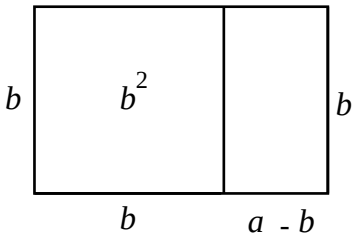
Devo determinare l'area di un rettangolo, conoscendo le sue dimensioni a e b:



In questo caso i passi da compiere sono:

1] scrivo la formula per determinare l'area del rettangolo: $A = a \cdot b$;
2] sostituisco ad *a* e a *b* i dati numerici;
3] moltiplico i valori numerici di *a* e di *b* (moltiplicando indifferentemente *a per b* oppure *b per a*);
4] scrivo il valore ottenuto.

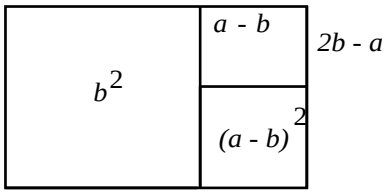
Anche in questo caso possiamo affermare che l'algoritmo usato è *efficace*, in quanto si conclude in un tempo finito, non conduce ad alcuna contraddizione, permette di applicarlo a problemi simili, permettendomi di risolverli. Ma non è il solo algoritmo che mi permetta di risolvere questo problema. Esso si potrebbe risolvere, per esempio, nel modo che segue.
Considero il rettangolo dato come ottenuto dalla somma di un quadrato di lato *b* e di un rettangolo di lati *b* e *a - b*:



per cui la sua area sarà:

$$A = b^2 + b\,(a - b)$$

Nello stesso modo, l'area del rettangolo ottenuto sarà data dalla somma di un quadrato il cui lato è *a - b* e di un rettangolo di lati *a - b* e *2b - a*:



$$A = b^2 + (a - b)^2 + (a - b)\,(2b - a)$$

Continuando nello stesso modo, otterremmo per l'area del rettangolo, una successione di quadrati e di rettangoli che, in linea di principio, non avrà mai fine, o, quantomeno, che mi farà ottenere il risultato mediante una lunga successione di operazioni, per cui l'algoritmo usato non potrà ritenersi *buono*.

h] Consideriamo ora il seguente problema:

Ottenere il numero 13 iniziando dal numero zero.

Supponiamo di volere risolvere il problema mediante il seguente algoritmo:

1] attribuisco ad *A* il valore *0*;

2] aggiungo *2* ad *A*;

3] scrivo il valore di *A*;

4] ripeto i passi *2* e *3* finché *A* diventa uguale a *13*.

Eseguendo le istruzioni, questo algoritmo fornisce la successione di valori:

0, 2, 4, 6, 8, 10, 12, 14, 16, . . .

Esso dunque non mi risolve il problema né ha termine, per cui non è un buon algoritmo.

i] Ecco un altro problema:
Debbo spiegare ad un amico come arrivare a casa mia, partendo da un posto noto ad entrambi.

Dopo avergli disegnato una piantina sommaria del luogo da raggiungere, potrei fornirgli queste indicazioni:

1] partendo dal punto noto, prendi la prima strada a destra;
2] prosegui lungo questa via fino alla seconda trasversale;
3] gira a sinistra;
4] prosegui fino a che vedi sulla tua destra una bella insegna di negozio;
5] gira a sinistra;
6] prosegui lungo questa via fino al n. 33.

A prima vista, sembra che la successione delle istruzioni vada bene, e che l'algoritmo debba funzionare. In realtà c'è qualcosa che non lo rende del tutto efficace: precisamente il passo 4, quando viene detto all'amico di proseguire finché non vede una bella insegna! In tal caso ho ipotizzato che quella che per me è una bella insegna debba esserlo pure per il mio amico! Ciò che può anche non essere, perché il concetto di “bello” è soggettivo. Quindi, anche questo algoritmo non è buono.

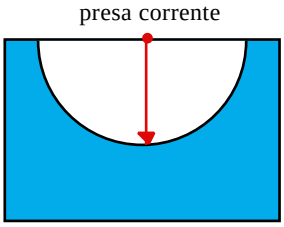
l] Prendiamo in considerazione un altro problema, che può essere proposto, per esempio, a ragazzi di una terza classe di scuola media:

Una stanza rettangolare misura 32 m di lunghezza e 18 m di larghezza.
Una presa di corrente si trova a metà di uno dei lati maggiori e all'altezza del suolo. Per pulire la stanza si usa un aspirapolvere che ha il cordone elettrico lungo 15 m.
Quanto misura la superficie che non si riesce a pulire?

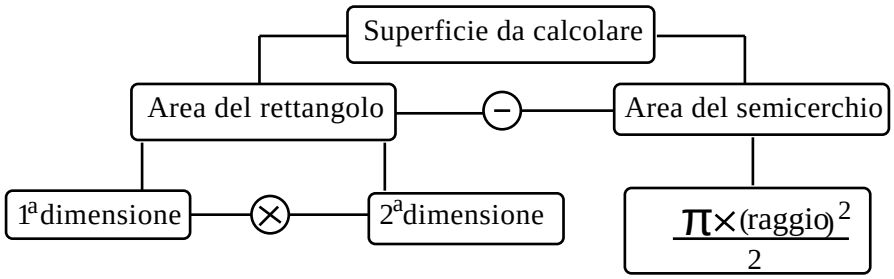
Supponiamo che non tutti i ragazzi riescano a risolvere il problema. Cerchiamo allora di formularlo diversamente, precisando meglio le parti che bisogna risolvere per giungere alla sua soluzione; cioè, cerchiamo di suddividerne le varie fasi che possono farlo risolvere in maniera più dettagliata.

Una stanza rettangolare misura 32 m di lunghezza e 18 m di larghezza.
Quanto misura la superficie della stanza?
Per pulirla si usa un aspirapolvere che ha il cordone elettrico lungo 15 m. Sapendo che c'è una presa di corrente a metà del lato più lungo e all'altezza del suolo, quanto misura la superficie che non si riesce a pulire?

Un disegno che illustra il problema conduce, questa volta, rapidamente alla soluzione:



Essa può essere schematizzata mediante un *diagramma ad albero*:



L'utilità dell'albero consiste nel fatto che esso permette di risolvere problemi analoghi a quello proposto, ed è utile per codificare l'algoritmo necessario per risolvere il problema, nel senso che si giunge alla soluzione procedendo in senso inverso, dal basso verso l'alto:

1] calcolo l'area del rettangolo, conoscendone le dimensioni;
2] calcolo l'area del semicerchio, conoscendo il raggio d'azione dell'aspirapolvere (lunghezza del cordone elettrico);
3] calcolo la differenza tra le due aree;
4] scrivo il risultato finale.

Mediante questa serie di indicazioni, gli studenti possono risolvere facilmente il problema, perché l'algoritmo che abbiamo costruito è:

- preciso e dettagliato;
- finito;
- univoco.

Quindi, come già s'è detto, *ogni algoritmo è legato strettamente alla risoluzione di un dato problema*.

Che cosa c'è realmente alla base della costruzione di un algoritmo?

In base a quanto detto, si pone la domanda: che cosa c'è alla base di un algoritmo? Per rispondere alla domanda bisogna analizzare meglio il passaggio dal problema alla sua soluzione.
Innanzitutto, è bene fare rilevare agli allievi che alla parola «problema» non si deve attribuire, come spesso accade, il significato restrittivo di «questione matematica da risolvere», bensì, come è stato mostrato dagli esempi precedenti, e rifacendosi all'originario significato del termine greco «προβλημα», quello di «situazione problematica» per la quale si cerca una soluzione.
Per sottolineare ciò, e per fare riconoscere agli allievi gli elementi e le caratteristiche che possono avere in comune, ecco un elenco di problemi di varia natura e complessità:

a) Mettere in ordine alfabetico un elenco di nomi.
b) Preparare le triglie alla livornese.
c) Determinare le dimensioni del modellino di piramide più grande possibile, a base quadrata, che si può ricavare da una sferetta di rame di raggio 15 cm.
d) È vero o falso che la radice cubica di 2 è maggiore della radice nona di 10?

Si riconosce facilmente che nella risoluzione di questi problemi è necessario esaminare tre aspetti fondamentali:

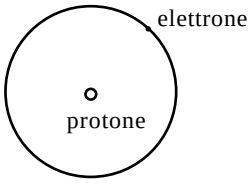
1. L'insieme dei dati di cui si dispone e l'insieme dei risultati che si ottengono.
2. L'insieme delle istruzioni che consentono di ottenere la soluzione del problema proposto.
3. Le capacità di chi deve eseguire le istruzioni.

Quindi, potremmo dire che *risolvere un problema significa ricercare ed esprimere una successione finita di istruzioni interpretabili da un esecutore, che conducano da determinate informazioni iniziali ad altre informazioni finali*.
Infatti, che cosa facciamo quando tentiamo di risolvere un problema? Ebbene, se ci riflettiamo un po', ci accorgiamo che procediamo in realtà sempre allo stesso modo: ci serviamo di un ***modello***, ovvero di ***una rappresentazione semplificata dello stesso problema, sfrondandolo di tutto ciò che è superfluo per raggiungere l'obiettivo (cioè la soluzione) che ci si propone.***

a) Se devo calcolare, per esempio, quanto nastro mi occorra per orlare una tovaglia quadrata il cui lato misura *1,4* metri, potrei risolvere il problema utilizzando un ***modello algebrico***, ovvero la formula per il calcolo del perimetro del quadrato di lato *l*:

$$p = 4 \cdot l$$

b) Se devo individuare il percorso stradale più breve per andare da Piazza Politeama a Piazza Magione, potrei utilizzare un ***modello descrittivo*** come lo stradario di Palermo o una cartina stradale della città.
3) Se devo spiegare ai miei alunni la struttura dell'atomo, posso utilizzare un ***modello descrittivo*** dell'atomo, qual è quello di Bohr.



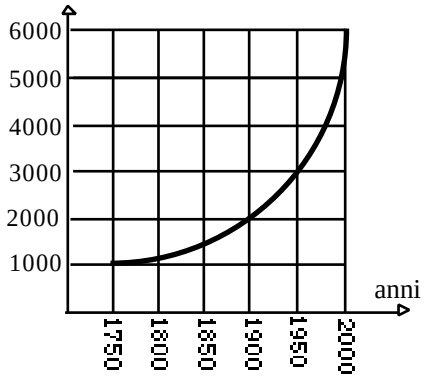
Modello di Bohr dell'atomo di Idrogeno

4) Se devo determinare la forza di attrazione gravitazionale tra la Terra e la Luna posso utilizzare il ***modello matematico*** elaborato da Newton:

$$F = G\frac{m\,M}{r^2}$$

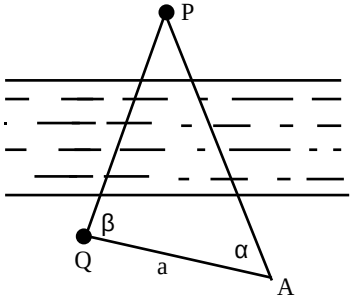
in cui F è la forza di attrazione gravitazionale, G è la costante di gravitazione universale, m è la massa della Luna, M è la massa della Terra ed r la distanza media fra i due corpi celesti.

5) Se devo prevedere la crescita della popolazione mondiale tra *10* anni, potrei utilizzare un ***modello predittivo*** come un grafico che mi indichi il numero di abitanti della Terra in funzione degli anni.



6) Se devo ridurre il mio peso fino ad un dato valore, posso utilizzare un ***modello prescrittivo*** come una dieta che mi indichi le quantità e le modalità di assunzione del cibo.

7) Se devo calcolare la distanza fra due zone inaccessibili utilizzerò un ***modello geometrico-trigonometrico***.



Il termine *modello* può quindi assumere una grande quantità di significati differenti l'uno dall'altro, secondo le situazioni. Ebbene, tra i modelli rivestono particolare interesse i ***modelli matematici***, che, «***sono rappresentazioni formali di idee o conoscenze relative ad un fenomeno espresse mediante il linguaggio matematico***" (G.Israel, *I modelli matematici*.)

Consideriamo, per esempio, il problema seguente:

Un foglio quadrato di carta dello spessore di un millimetro viene piegato 50 volte su se stesso. Determinare lo spessore che avrà il foglio dopo le 50 piegature.

È chiaro che se si vuole giungere alla soluzione di questo problema, non si potrà utilizzare un modello fisico, cioè un foglio di carta e cominciare a piegarlo, misurando ad ogni piegatura lo spessore raggiunto. Infatti, un modello del genere risulterebbe insufficiente e inadeguato per il problema posto, in quanto ad un certo punto il foglio non si potrebbe più piegare su se stesso.

È necessario, quindi, abbandonare quest'idea e utilizzare invece un *modello matematico* che sia adeguato a descrivere la situazione reale presa in esame. Cerchiamo, quindi, di ragionare: il foglio dopo la prima piega avrà uno spessore di 2 mm; dopo la seconda piega lo spessore sarà di 4 mm, perché abbiamo piegato su se stesso un foglio dello spessore di 2 mm. Dopo la terza piega lo spessore del foglio sarà allora di 8 mm, dopo la quarta piega sarà di 16 mm, e così via.
Notiamo, dunque, una certa regolarità tra il numero delle pieghe e lo spessore che via via il foglio va assumendo. Possiamo, allora, riassumere i risultati in uno schema;

Spessore iniziale	*Numero delle pieghe*	*Spessore finale*
1 mm	0	1 mm
1 mm	1	2 mm
2 mm	2	4 mm
4 mm	3	8 mm
8 mm	4	16 mm
16 mm	5	32 mm

Dall'osservazione dei dati si può assumere come modello matematico l'equazione

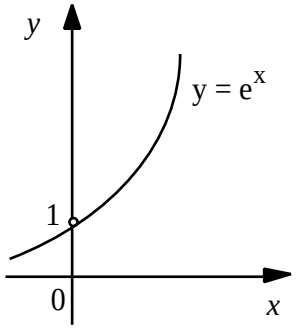
$$spessore\ finale = 2^{\,numero\ delle\ pieghe}$$

Ponendo y = spessore finale e x = numero delle piegature, si ottiene la funzione

$$y = 2^x$$

il cui valore, per $x = 50$ è:

$$y\,(50) = 2^{50} \sim 1{,}13 \cdot 10^{15}\ \text{mm} = 1{,}13 \cdot 10^{9}\ \text{km}.$$



Quindi, lo spessore del foglio è pari a circa *1,13* miliardi di chilometri! Per comprendere meglio questo risultato, si pensi che, per esempio, la distanza Terra-Sole è circa *150* milioni di chilometri, mentre la distanza Sole-Plutone, che è l'ultimo pianeta del sistema solare, è di *5,91* miliardi di chilometri. Ciò significa che il nostro foglio, dopo le *50* piegature, avrà uno spessore pari alle dimensioni del sistema solare!
A questo risultato non saremmo potuti giungere senza l'aiuto del modello matematico, che ci ha permesso di risolvere il problema, la cui soluzione non sarebbe stata comprensibile se si fosse utilizzata soltanto l'esperienza sensibile.

Possiamo affermare, quindi, che servirsi di un modello per affrontare la risoluzione di un problema è realmente utile, perché il modello ci permette di rappresentare il problema in modo chiaro, sintetico e privo di informazioni superflue; inoltre, una volta stabilito, esso é efficace per chi lo usa perché diventa una fonte di informazioni.
In realtà, la ***modellizzazione di un problema*** è una fase di estrema importanza, perché se essa non si realizza in maniera adeguata, non permetterà di raggiungere gli obiettivi che si sono

prefissati. Bisogna tenere anche presente che un modello non ha una validità assoluta, perché questa viene limitata dalle condizioni di applicabilità dello stesso modello.
Infine, se un dato problema non ammette soluzione in una certa modellizzazione, può ben darsi che l'ammetta invece in un'altra.
Si deve quindi far comprendere agli allievi che i modelli non sono entità *statiche*, bensì *dinamiche*, e una buona pratica didattica consisterà nel fare in modo che essi, per un dato problema, vengano sollecitati a trovare diversi modelli che possano condurre alla soluzione; scegliendo, poi, quello che comporta la soluzione più semplice.
Dunque, quando si è saputo costruire un modello, è stato fatto un passo avanti nella risoluzione di un problema, ma è proprio il passaggio dal problema al modello e da questo alla soluzione del problema che a volte non è tanto immediato.

Infatti, una volta costruito il modello, occorre pensare a tutta una serie di operazioni da compiere per giungere alla soluzione di un problema.
Così, per esempio, avere disegnato una figura geometrica per risolvere un problema di geometria non significa che il problema sia già stato risolto. Il modello è solo necessario per individuare una strategia per risolvere il problema.
Ebbene, in ogni situazione problematica, sia essa di natura matematica o no, riscontriamo degli elementi comuni:
a) una ***fase iniziale*** in cui ci vengono fornite delle informazioni iniziali, che sono i ***dati*** del problema, e le relazioni fra di essi;

b) una ***fase principale*** in cui si scompone il problema in sottoproblemi, (detti anche ***stati***) e si costruisce, sulla base del modello adottato, una ***procedura di soluzione***, elaborando i dati mediante le relazioni note, e che ci permette di ricavare nuove informazioni.

c) una ***fase finale*** in cui attraverso l'elaborazione di tutte le informazioni, quelle date e quelle ricavate, giungiamo alla soluzione del problema, ovvero al ***risultato***. Durante questa fase vi è una transizione di stati, si passa, cioè, da uno stato all'altro per giungere alla soluzione.

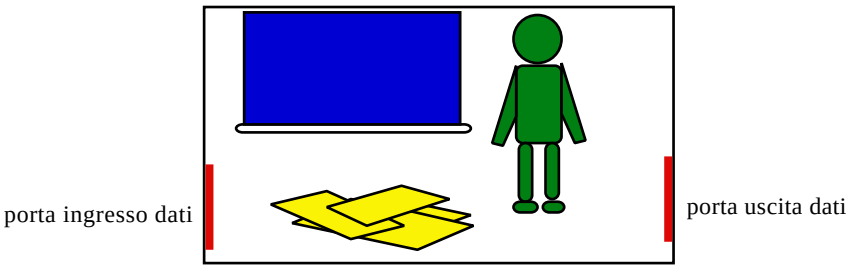
Il *punto nodale*, nella risoluzione di un problema, è, quindi, proprio la costruzione di una procedura risolutiva, ovvero di una sequenza ordinata di azioni *effettivamente eseguibili*, ovvero di ciò che abbiamo chiamato ***algoritmo***.
E la descrizione di un algoritmo sarà corretta quando alle successive istruzioni elencate corrisponderanno le transizioni di stato desiderate, fino a giungere allo stato finale, ovvero alla soluzione.
Ecco, quindi, la risposta che possiamo dare alla domanda iniziale: *alla base della costruzione di un algoritmo vi è la necessità di rendere operativo il modello adottato per risolvere il problema*.
Per utilizzare un algoritmo è necessario che esso venga eseguito, cioè che ci sia un *esecutore.*
L'*esecutore*, può essere o la stessa persona che lo ha costruito o un'altra persona o anche un *automa esecutore.*
Se, per esempio, l'esecutore di un dato algoritmo è uno studente, egli lo esegue *autonomamente*, cioè ad ogni passo comprende quel che sta facendo e perché lo fa.
Ma quando l'esecutore di un algoritmo, ovvero "chi" lo applica per risolvere un dato problema, è una macchina, come, per esempio, un computer, allora la sequenza delle istruzioni dev'essere il più possibile precisa e dettagliata, ordinata e finita, perché tali istruzioni, tradotte mediante un *linguaggio di programmazione*, devono potere essere eseguite *alla cieca*, cioè,

senza che l'esecutore (in questo caso il computer) capisca che cosa stia facendo, proprio perché esso non ha coscienza di ciò che fa, non essendo intelligente.
Come ha scritto Roger Penrose in *La mente nuova dell'imperatore* (p. 40):

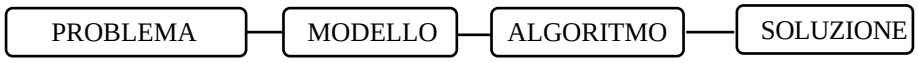
Un algoritmo che ambisse a eguagliare quello che si presume operi in un cervello umano dovrebbe essere di necessità una cosa prodigiosa.

Il computer ha la ***sola capacità*** di eseguire le istruzioni che gli sono state impartite, attraverso il linguaggio di programmazione, in maniera rapida, sicura e affidabile, ma non autonomamente.
In altre parole, mentre un problema può essere proposto ad uno studente, e attendere che egli lo risolva anche da solo, nel senso che può prendere delle iniziative (essendo intelligente), *ciò non si può pretendere da un computer, in quanto bisogna suggerirgli passo dopo passo le azioni da compiere per giungere alla soluzione.*
Volendo rappresentare le due situazioni, cioè quando l'esecutore dell'algoritmo è l'uomo oppure il computer, possiamo ottenere la situazione seguente:

Volendo rappresentarci l'esecutore del computer, potremmo raffigurarlo come un omino, chiuso dentro una stanza in cui ha a disposizione solo una lavagna, del gesso, un blocco di carta e una penna. L'omino non ha alcun contatto con il mondo esterno se non attraverso due porte che possono aprirsi solo se ciò è previsto da una delle istruzioni dell'algoritmo.
Attraverso una porta vengono forniti all'omino dei dati; lui li elabora seguendo le istruzioni dell'algoritmo; trascrive i risultati su un foglio di carta e li passa attraverso la seconda porta.



Possiamo, dunque, schematizzare una situazione generale in cui si rende necessaria la costruzione di un algoritmo, nel modo che segue:



Ebbene:

Ogni volta che una successione di azioni si automatizza, allora si compie un vero passo avanti.

Se ciò è vero per tanti piccoli problemi della vita quotidiana, a maggior ragione varrà per i problemi che la scienza in generale e la matematica in particolare ci pone.
La costruzione di un algoritmo è quindi necessaria per tradurre il modello utilizzato in un linguaggio che sia comprensibile per l'esecutore, sia esso uomo o macchina. Il linguaggio

utilizzato può essere anche quello che usiamo normalmente nella vita di tutti i giorni. Ma, come sappiamo, spesso, il *linguaggio dei simboli è più chiaro e più pratico da utilizzare*.
Se, per esempio, sul cruscotto della nostra macchina non fossero disegnati dei simboli adatti a segnalarci, attraverso delle spie luminose, che dobbiamo fare benzina, o che la pressione dell'olio non è sufficiente, noi non potremmo prendere delle decisioni in tempo utile.
Analogamente, un segnale stradale può anche trasmetterci, con pochi simboli, delle informazioni abbastanza complesse, come, per esempio, il divieto di parcheggiare in un certo luogo o quello di usare il clacson, quando, per esempio, si transita davanti ad un ospedale.
Si pone, quindi, il problema di usare, anche per gli algoritmi, una rappresentazione schematico-simbolica che ne descriva chiaramente la struttura logica.

Struttura generale di un algoritmo

Per delineare la struttura generale di un algoritmo, riprendiamo l'esempio dell'algoritmo del prodotto *p* di due interi arbitrari non negativi *a* e *b*, trovato nel Papiro Rhind e riportato all'inizio di questa parte.

Algoritmo MOLTIPLICAZIONE *(**dati** a e b, **risultato** p):*
__poni__ p = 0;
__finché__ a ≠ 0 __ripeti__ la sequenza:
__se__ a è dispari __allora__ addizione b a p;
dimezza a trascurando il resto;
raddoppia b.

L'algoritmo, come si nota, è stato scritto con dei *costrutti* che, come si vedrà, sono caratteristici dei linguaggi programmativi. Quali sono le motivazioni del loro impiego? Essenzialmente tre. La prima, è che questo è proprio il mezzo espositivo caratteristico dell'informatica; la seconda, è che questi costrutti consentono *concisione* e *chiarezza* che difficilmente possono essere ottenute in altro modo; la terza è che essi hanno carattere di *universalità*
I costrutti chiave dell'algoritmo sono indicati con le parole scritte in grassetto. Esaminiamole.
- La parola ***algoritmo***, di cui MOLTIPLICAZIONE è il nome scelto arbitrariamente, indica un meccanismo di calcolo definito una volta per tutte e individuabile per nome.
- L'algoritmo può essere utilizzato su ***dati*** arbitrari (in questo caso interi non negativi) che acquistano all'interno dell'algoritmo i nomi locali di *a* e *b*, e viene generato come ***risultato*** il prodotto dei dati con il nome *p*.
- Le frasi che descrivono l'algoritmo devono essere interpretate ed eseguite in ***sequenza*** nell'ordine in cui si leggono. Vi sono frasi che specificano operazioni aritmetiche come l'addizione, la divisione e la moltiplicazione per *2*; ma le frasi più importanti sono le *assegnazioni di valore:* ***esplicite*** come "***poni*** *p = 0*" o ***implicite*** come "*addiziona b a p*" (cioè, "*assegna a p il nuovo valore ottenuto addizionando il valore corrente di p a quello di b*). In un algoritmo le assegnazioni sono importanti perché la loro specificazione implica che l'algoritmo *incide su un mondo circostante*, di cui esso varia lo ***stato***, che qui è identificato da una corrispondenza tra ***nomi*** di oggetti e loro ***valori***.
Ebbene, *nella definizione semantica dei linguaggi programmativi, lo studio di queste implicazioni è cruciale.*
- I costrutti di ***iterazione:***

finché condizione ***ripeti*** una sequenza di passi

e di ***esecuzione condizionata:***

se condizione ***allora*** passo1 ***altrimenti*** passo2

sono fondamentali, perché (come si vedrà) sono sufficienti a specificare la struttura di controllo di qualsiasi algoritmo.
Inoltre, è bene chiarire la differenza tra un *algoritmo iterativo* e un *algoritmo ricorsivo*.
Un algoritmo *iterativo* è basato sull'esplicita ripetizione di una o più operazioni elementari, controllata da un meccanismo di arresto.
Un algoritmo *ricorsivo* richiama se stesso su sottoinsiemi dei dati e fornisce la soluzione finale combinando le soluzioni parziali ottenute dalla sua esecuzione sui sottoinsiemi.
Ora, in generale, qualunque problema può essere impostato con un algoritmo iterativo o ricorsivo, ma non esiste, in assoluto, un metodo migliore dell'altro.

Sull'efficacia di un algoritmo

Come già s'è detto, in generale, ci si aspetta che un algoritmo sia *efficace*, cioè che le sue operazioni siano sufficientemente fondate in modo da essere eseguibili, in linea di principio, da chiunque, usando carta e penna.
Consideriamo, per esempio, l'*algoritmo euclideo* (che dopo tratteremo più approfonditamente) per determinare il *massimo comun divisore* tra due numeri interi positivi m ed n ($m > n$), ovvero il più grande intero positivo che li divide entrambi. Esso può essere rappresentato nel modo seguente:

1. Si divida m per n, e sia r il resto ($0 \leq r < n$).

2. Se $r = 0$, l'algoritmo termina; la risposta è n.

3. Si ponga $m \leftarrow n$, $n \leftarrow r$, e si ripeta il passo 1. ($m \leftarrow n$ significa che si sostituisce m con n.)

Le operazioni da eseguire sono efficaci, innanzitutto perché i numeri interi si possono rappresentare su carta in forma finita, e poi perché vi è almeno un metodo per eseguire l'operazione di divisione. Ma le stesse operazioni non sarebbero più efficaci se, invece di interi positivi, noi considerassimo due qualsiasi numeri reali rappresentati da uno sviluppo decimale infinito, né se i valori fossero le lunghezze di segmenti rettilinei fisici (che non si possono specificare esattamente). Supponiamo, per esempio, di avere un algoritmo in cui uno dei passi prescriva la seguente operazione:

«Se 4 è il più grande intero n per cui esiste una soluzione dell'equazione $w^n + x^n + y^n = z^n$ in interi positivi w, x, y e z, allora vai al passo 4.»
Un'affermazione di questo genere non può mai essere efficace finché qualcuno non abbia costruito effettivamente un algoritmo per determinare se realmente 4 sia o no il più grande intero che goda della proprietà stabilita.
Come scrive Donald E. Knuth nel primo volume della sua opera: *The Art of Computer Programming* (p. 7):

«*In pratica, noi non vogliamo solo algoritmi, noi vogliamo algoritmi che siano buoni in un certo senso dal punto di vista estetico. Un criterio di bontà è la durata del tempo necessario per eseguire l'algoritmo; ciò può essere espresso attraverso il numero di volte in cui ciascun passo dev'essere eseguito. Altri criteri sono l'adattabilità dell'algoritmo a differenti specie di computer, la sua semplicità ed eleganza, ecc.*
Spesso abbiamo a disposizione parecchi algoritmi per lo stesso problema, e dobbiamo decidere quale sia il migliore. Ciò ci porta al campo estremamente interessante e molto importante dell'analisi algoritmica: Dato un algoritmo, se ne vogliono determinare le caratteristiche di esecuzione.»

Consideriamo, per esempio, l'algoritmo euclideo da questo punto di vista, e supponiamo di chiederci: supponendo che, tra i due numeri m ed n, sia noto n, mentre m può variare su tutti i valori interi positivi, qual è il numero medio di volte T_n, che il passo 1 dell'algoritmo dovrà essere eseguito?
Ebbene, la questione importante è determinare la natura di T_n; esso è, per esempio, approssimativamente uguale a $\frac{1}{3}n$ o a $\sqrt{n}$? Ciò costituisce un problema matematico

estremamente interessante ma difficile, che non è stato ancora risolto del tutto. Si può però dimostrare che per grandi valori di n, T_n è approssimativamente uguale a:

$$\frac{12\,(\ln 2)}{\pi^2}\ln n$$

cioè, esso è proporzionale al logaritmo naturale di n.
In generale, si può dire che la maggior parte degli algoritmi ha un parametro principale N, che di solito è dato dal numero dei dati messi in gioco, e che incidono significativamente sul tempo di esecuzione dell'algoritmo. Il parametro N può essere il grado di un certo polinomio, la grandezza di un file, il numero dei nodi in un grafo, e così via. Virtualmente, il tempo di esecuzione di un algoritmo può essere proporzionale:
- ad 1, e allora viene detto *costante*;
- a $\log N$, per cui viene detto *logaritmico*, e in questo caso il programma scorre più lentamente di quanto cresca N;
- ad N, e allora viene detto *lineare*;
- a $N \log N$, e questo è il caso di algoritmi che risolvono un problema, spezzettandolo in sottoproblemi più piccoli, risolvendoli indipendentemente, e combinando poi le soluzioni;
- a N^2, allora viene detto *quadratico*, e l'algoritmo si può applicare solo a piccoli problemi;
- a N^3, allora viene detto cubico, e anche in questo caso l'algoritmo si può applicare solo a piccoli problemi;
- 2^N, e viene detto *esponenziale*. Pochi algoritmi con un tempo d'esecuzione esponenziale hanno un'effettiva applicazione, perché il loro tempo di esecuzione si dilata paurosamente all'aumentare delle dimensioni dei dati.

Generalità sulla rappresentazione grafica degli algoritmi

Perché mai, o Dei, due e due dovrebbe fare quattro?

Alexander Pope (1688-1744)

Alla base della costruzione e della rappresentazione degli algoritmi c'è, innanzitutto, la comprensione del significato di. alcuni termini che vengono usati frequentemente.

- ***Dati***: sono tutte quelle conoscenze che caratterizzano una particolare situazione problematica e che possono anche essere conservati per essere utilizzati in futuro.
I dati su cui un algoritmo opera si distinguono in:

a) Dati *iniziali*, cioè quei dati che vengono assegnati quando si formula un problema;

b) Dati *intermedi,* cioè quelli che si ottengono durante l'elaborazione e che vengono usati in altre fasi del procedimento risolutivo;

c) Dati *finali* o *risultati*, cioè quelli ottenuti dall'elaborazione dei dati iniziali e intermedi e che forniscono la soluzione del problema.

Inoltre, i valori dei dati possono essere:

a) *numerici*: per esempio i valori *34, 5, 89, ...* ;

b) *logici*: vero, falso;

c) *alfanumerici*: «*2* aprile», « è maggiore», ...

- ***Elaborazione***: è il trattamento dei dati, dalla loro semplice trascrizione su un foglio diu carta fino alla loro manipolazione nei calcoli matematici più complicati. L'elaborazione dei dati può dare luogo ad altri dati che si aggiungono a quelli noti per contribuire alla soluzione del problema.

- ***Istruzione***: ogni azione che l'esecutore deve compiere, che può essere, per esempio, un calcolo da fare, oppure una semplice acquisizione di dati, e così via.

- ***Variabile***: una qualsiasi quantità che può essere modificata, e assumere diversi valori.

- ***Costante***: una qualsiasi quantità che non muta il proprio valore, come, per esempio, il numero $\pi = 3{,}1415927$ ...
I valori che si assegnano alle variabili o alle costanti si indicano, di solito, con le lettere dell'alfabeto minuscolo: *a, b, c, . . . , x, y, . . .*
Molto importante è l'*assegnazione* di un valore ad una variabile. Se, per esempio, alla variabile *m* viene assegnato il valore *p,* ciò si suole rappresentare con la scrittura:

$$m \leftarrow p$$

e si legge: «il valore *p* viene assegnato ad *m*», oppure: «il valore della variabile *m* viene sostituito dal valore della variabile *p*». Ad esempio, l'operazione di aumentare il numero *n* di 1 viene denotata con la scrittura: $n \leftarrow n + 1$.
Non si deve confondere il simbolo di assegnazione con il simbolo =. Infatti, la scrittura $x=a$ dichiara che l'oggetto *x* ha lo stesso valore dell'oggetto *a*, e tale dichiarazione può essere sia vera che falsa, mentre la scrittura $x \leftarrow a$ assegna all'oggetto *x* lo stesso valore dell'oggetto *a*, e non ha nemmeno senso domandarsi se la cosa è vera o falsa, poiché è vera per definizione. Analogamente, mentre dal punto di vista matematico la scrittura $x = x + 1$ è evidentemente errata, la formula di assegnazione $x \leftarrow x + 1$ è perfettamente logica in quanto traduce l'espressione «assegna ad *x* il valore attuale di *x* più uno». È dunque importante far comprendere agli allievi che ogni assegnazione del tipo $y \leftarrow nuovo$, fa assumere all'oggetto *y* indicato dalla freccia il valore «*nuovo*» e contemporaneamente y perde il valore che aveva precedentemente.

- ***Sequenza***: una successione di istruzioni da eseguire in un ordine stabilito.

- ***Selezione binaria***: la scelta fra due possibili alternative. Per esempio, se devo calcolare la radice cubica di un numero, essa sarà positiva o nulla se altrettanto lo è il numero, altrimenti sarà negativa.

- ***Iterazione***: una serie di indicazioni che devono essere ripetute nello stesso ordine un certo numero di volte.

- ***Sistema***: un qualsiasi insieme di elementi di natura diversa che interagiscono in modo dinamico fra di loro per raggiungere un obiettivo. Esso può essere ***chiuso***, se è isolato, cioè, quando non interagisce con l'ambiente esterno; ***aperto*** in caso contrario.

Un algoritmo si rappresenta essenzialmente mediante tre tecniche:

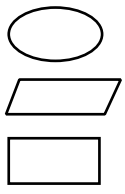
- il ***diagramma di flusso***;

- i ***grafi Nassi-Schneidermann*** (abbreviati con la sigla GNS);

- gli ***pseudolinguaggi***.

1. *Diagramma di flusso*

È stata una delle prime tecniche per rappresentare un algoritmo, risalente ai lavori pionieristici del grande matematico americano di origine ungherese John von Neumann (1903-1957).
Oggi, i diagrammi di flusso vengono utilizzati quasi esclusivamente per rappresentare parti molto limitate di algoritmi, a solo scopo esplicativo.
Comunque, didatticamente la loro validità è indubbia, perché aiutano ad organizzare la struttura logica dell'algoritmo. Un diagramma di flusso viene anche detto *flow-chart*, secondo la terminologia inglese.
Per costruire un diagramma di flusso si usano dei simboli ormai accettati universalmente, di cui alcuni di uso più frequente sono i seguenti:



Simbolo che indica l'***inizio*** o la ***fine*** di un algoritmo

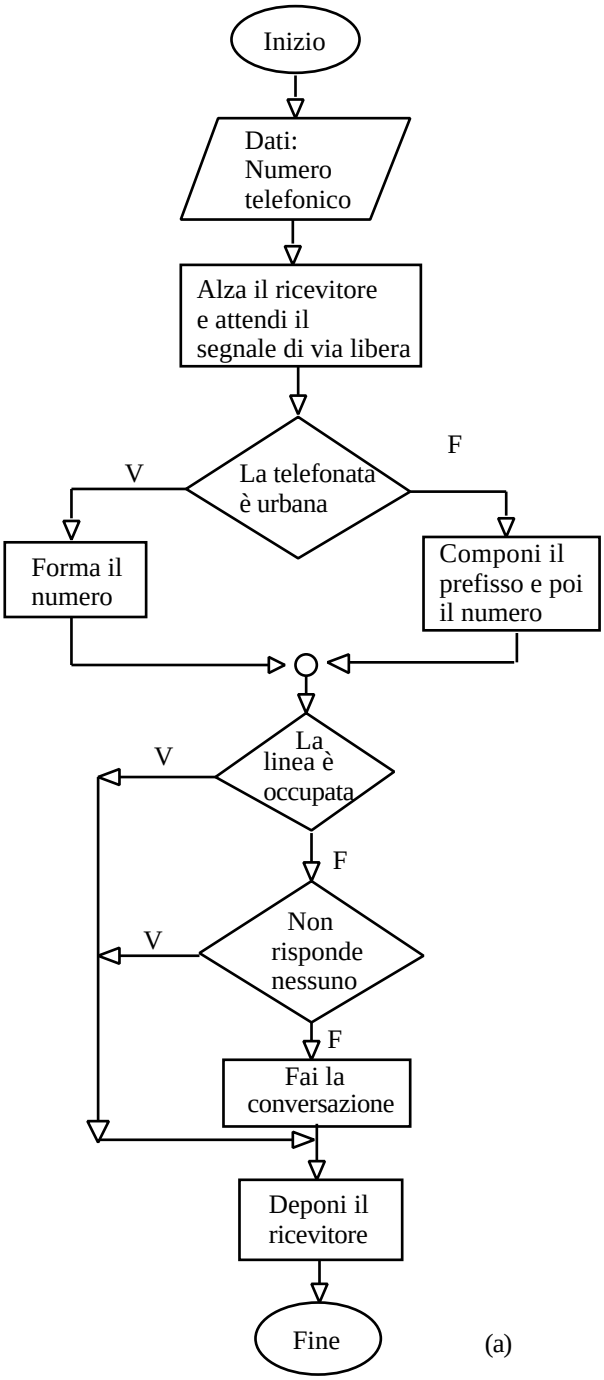
Simbolo che indica l'***immissione*** o l'***emissione*** di dati

Simbolo che indica l'***elaborazione*** di dati

Simbolo che indica la ***selezione***

Mostriamo, brevemente, alcuni esempi di come si utilizzano questi simboli.

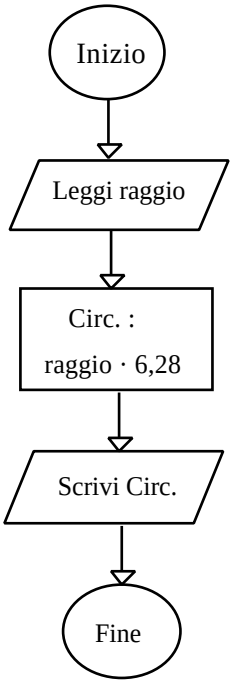
a) ***Rappresentazione di un algoritmo per fare una telefonata.***



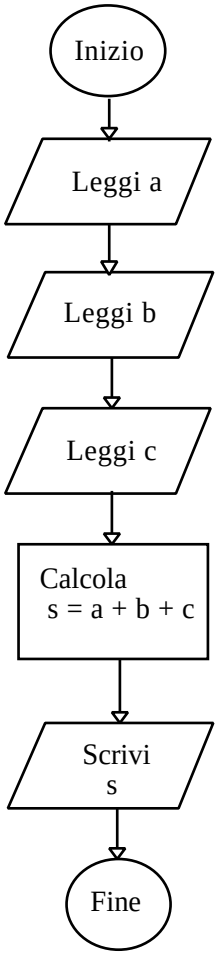
(a)

b) ***Rappresentazione di un algoritmo per calcolare la lunghezza di una circonferenza***.

c) ***Rappresentazione di un algoritmo per calcolare la somma di tre numeri***.



(b)



(c)

2. *I grafi Nassi-Schneidermann*

Questi grafici prendono il nome dagli studiosi *Nassi* e *Schneidermann* che per primi li proposero, e sono disegnati come una serie di scatole, una dentro l'altra, all'interno delle quali vengono descritte le operazioni da eseguire.
I simboli usati rappresentano:

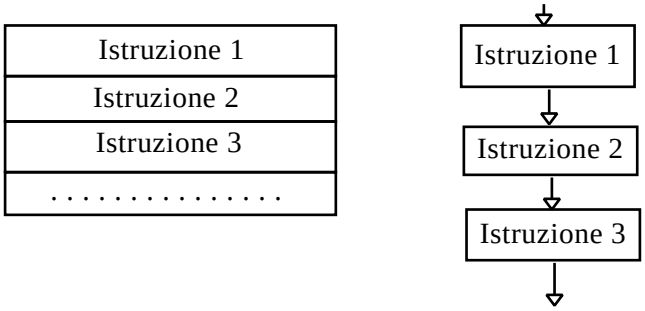
a) la sequenza

b) la selezione

c) l'iterazione.

a) ***La Sequenza***

È la più semplice delle strutture di controllo di un algoritmo. La sua sintassi consiste nell'elencare le azioni da eseguire una dopo l'altra, in successione ordinata, senza possibilità di scelta. Accanto alla sua rappresentazione si dà pure lo schema corrispondente usato nel diagramma di flusso:

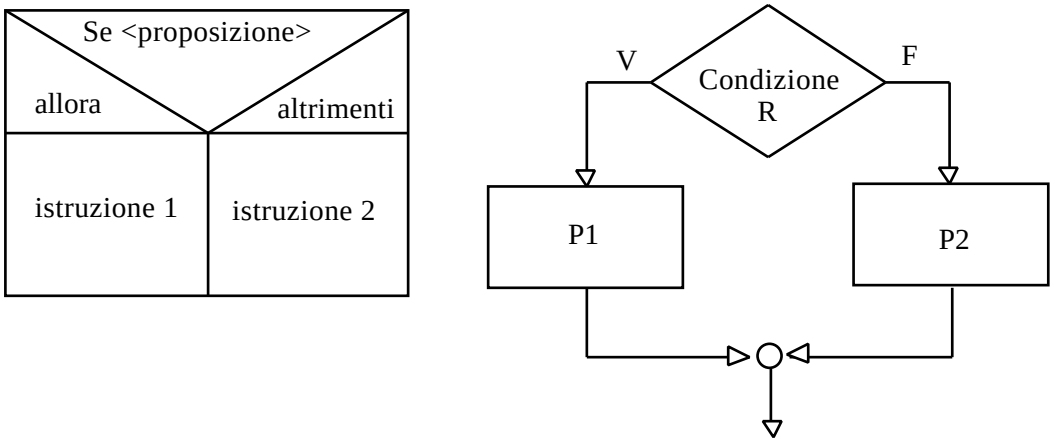


Ecco un esempio per calcolare la media fra tre numeri:

Leggi a, b, c
Somma : = a + b + c
Media : = Somma / 3
Scrivi Media

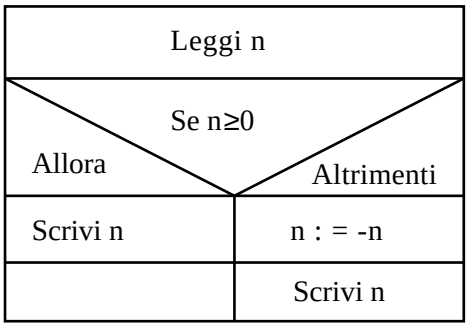
b) ***La Selezione binaria***

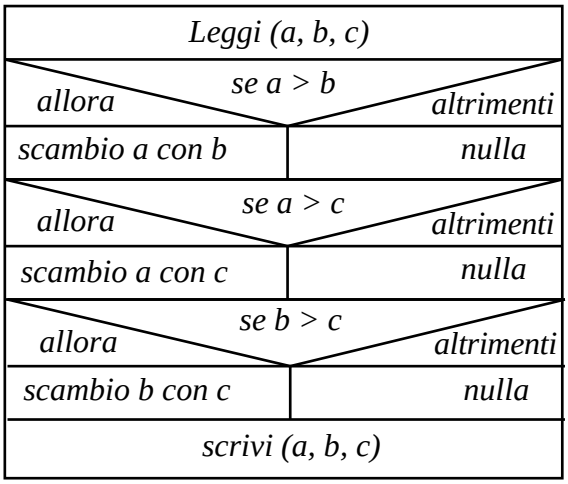
Viene rappresentata da una scatola divisa in due zone, una per scrivervi le istruzioni da eseguire nel caso che la proposizione posta sia vera; l'altra per scrivervi le istruzioni da eseguire in caso di falsità. Accanto alla sua rappresentazione si dà pure lo schema corrispondente usato nel diagramma di flusso:



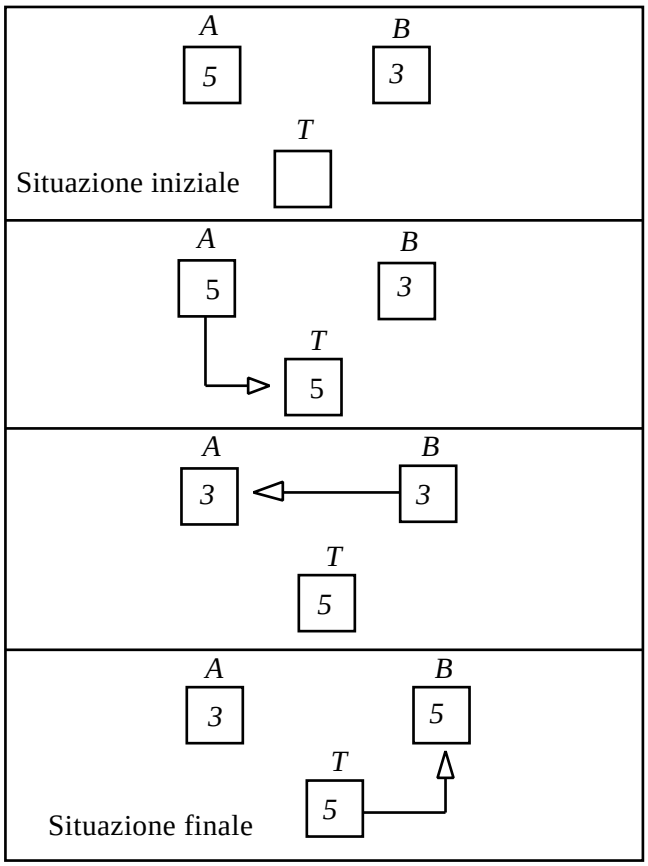
«Se R è vero allora fai P1,
altrimenti fai P2.»

Ecco un esempio per il calcolo del valore assoluto di un numero e per ordinare in senso crescente tre numeri:





A proposito della *procedura di scambio* presente nel secondo algoritmo, è bene tenere presente che per la sua esecuzione è necessario usare una terza variabile *T*, che serve da deposito temporaneo di uno dei due numeri. Così, se bisogna scambiare *5* (che si trova nella scatola *A*) con *3* (che si trova nella scatola *B*) per prima cosa *5* viene memorizzato in *T*, poi *3* in *A* e infine *T* (che ha il vecchio valore di *A*) in *B*, secondo lo schema seguente:



c) ***L'Iterazione***

È la struttura di controllo che viene usata quando un'istruzione o un gruppo di istruzioni devono essere ripetute finché non si verifichi una determinata condizione.

Può essere rappresentata mediante tre scatole secondo che si tratti di:

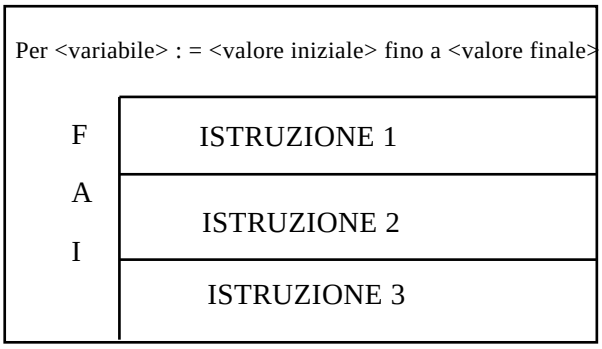
i) *iterazione enumerativa*;

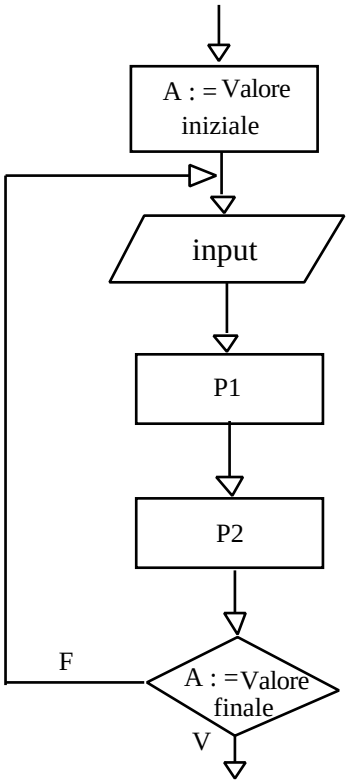
ii) *iterazione per vero*;

iii) *iterazione per falso*.

i) Iterazione enumerativa

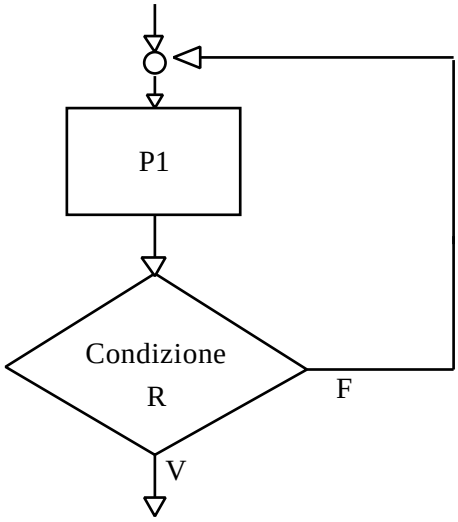
Si chiama così perché *enumera* le istruzioni che devono essere ripetute per raggiungere un dato scopo.
La sua sintassi è esemplificata dallo schema seguente, accanto al quale viene dato pure lo schema corrispondente usato nel diagramma di flusso:





ii) Iterazione per falso

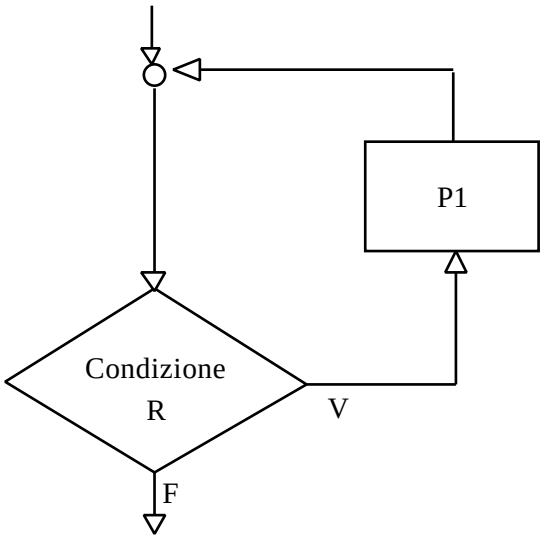
È una struttura di controllo in cui le istruzioni vengono ripetute in successione finché la proposizione analizzata risulta *falsa*; l'uscita dal ciclo avviene quando la proposizione diventa vera.
La sua sintassi è esemplificata dallo schema seguente, accanto al quale viene dato pure lo schema corrispondente usato nel diagramma di flusso:



«Finché R non è vera ripeti P1»

iii) Iterazione per vero

È la struttura di controllo in cui prima viene eseguita l'analisi della proposizione e poi, *solo se è vera*, vengono eseguite le istruzioni.
La sua sintassi è esemplificata dallo schema seguente, accanto al quale viene dato pure lo schema corrispondente usato nel diagramma di flusso:



«mentre R è vero ripeti P1»

È bene tenere presente, a proposito di queste tre strutture di controllo, che nella programmazione strutturata (in cui un algoritmo si rappresenta mediante una serie di blocchi, ciascuno isolato dagli altri e con una sola entrata e una sola uscita) si dimostra il ***teorema di Böhm-Jacopini:***

È sempre possibile scrivere un algoritmo facendo uso di tre sole strutture di controllo: la sequenza, la selezione, l'iterazione.

3. *Gli pseudo-linguaggi*

Sono costituiti da espressioni verbali vicine al nostro modo di esprimerci, che descrivono le operazioni da compiere.
Ogni istruzione compiuta termina con il simbolo";" tranne l'ultima che termina con il punto".", per indicare che l'algoritmo è finito. Rappresentiamo, per esempio, in ***linguaggio di progetto*** (che è lo pseudo-linguaggio più diffuso) due algoritmi: il primo che calcoli la somma dei primi 20 numeri naturali; e il secondo che calcoli la media di tre numeri.

I° algoritmo

```
Inizio
    azzera la variabile SOMMA;
    azzera la variabile NUMERO;
    ripeti le istruzioni
        incrementa la variabile NUMERO di 1 unità;
        addiziona NUMERO a SOMMA;
    finché NUMERO = 20;
comunica il valore assunto dalla variabile SOMMA;
fine.
```

II° algoritmo

```
Inizio
    leggi (a, b, c);
    SOMMA : = a + b + c;
MEDIA : = SOMMA/3;
    scrivi (MEDIA);
fine.
```

III° algoritmo per moltiplicare due numeri (moltiplicazione egiziana o dei contadini russi

Riprendiamo in esame la moltiplicazione di due numeri secondo il *Papiro Rhind*. Osserviamo innanzitutto che:
- quando abbiamo un numero scritto in base 2, cioè mediante potenze del 2, e vogliamo ottenere il suo doppio non dovremo fare altro che aggiungere uno zero in coda all'allineamento binario (come, equivalentemente, facciamo quando moltiplichiamo un numero in base 10 proprio per 10).

Esempio

$a = 1\ 1\ 1\ 0\ 0\ 1$ $\quad [a = (57)_{10}]$
$2 \cdot a = 1\ 1\ 1\ 0\ 0\ 1\ 0$ $\quad [2 \cdot a = (114)_{10}]$

- Quando dobbiamo dividere per 2 un numero scritto in base 2, non dobbiamo fare altro che sopprimere l'ultima cifra a destra dell'allineamento, cifra che rappresenta il resto della divisione (così come facciamo con i numeri in notazione decimale).

Esempio

$a = 1\ 1\ 1\ 0\ 0\ 1$ $\quad [a = (57)_{10}]$
$\frac{a}{2} = 1\ 1\ 1\ 0\ 0$ $\quad \left[\frac{a}{2} = (28)_{10}\text{, resto} = 1\right]$

Si possono quindi considerare le operazioni

n **div** 2	che fornisce il quoziente della divisione esatta di n per 2;
n **mod** 2	che fornisce il resto della divisione esatta di n per 2;
$2 \cdot n$	che fornisce il doppio di n.

Ciò premesso, consideriamo il numero n scritto in base 2:

$$n = (a_k\, a_{k-1} \ldots a_1\, a_0)_2$$

dove ciascun a_h, $0 \le h < k$, vale *0* oppure *1*, mentre $a_k = 1$. Dunque:

$$n = 2^k + a_{k-1}\, 2^{k-1} \ldots a_1\, 2 + a_0$$

Il prodotto $n \cdot m$ sarà allora:

$$n \cdot m = 2^k\, m + a_{k-1}\, 2^{k-1} m \ldots a_1\, 2\, m + a_0\, m$$

Quindi, in definitiva, il prodotto $n \cdot m$ può essere espresso come somma di addendi del tipo

$$2^h\, m \qquad \text{con } h = 0, 1, 2, \ldots, k$$

D'altra parte, la successione dei valori

$$m, 2m, \ldots, 2^{k-1} m, 2^k\, m$$

si ottiene a partire da *m* mediante ripetuti raddoppi. Possiamo quindi scrivere il seguente algoritmo in *linguaggio di progetto*:

0. $N \leftarrow n, M \leftarrow m$

1. $P \leftarrow 0$
2. *finché N > 0, ripetere:*
2.1 *se N mod 2 = 0,*
allora:

2.1.1 $N \leftarrow N$ *div 2*

2.1.2 $M \leftarrow 2 \cdot M$
altrimenti:

2.1.3 $N \leftarrow N - 1$

2.1.4 $P \leftarrow P + M$
3. *stampare P*
4. *fine*

Ogni volta che vogliamo fare eseguire un algoritmo ad un automa (computer) dobbiamo aggregare le azioni elementari che esso è in grado di eseguire, secondo le tre strutture di controllo di cui abbiamo parlato prima, perché il teorema di Böhm e Jacopini ci assicura che con questi schemi possiamo costruire *qualsiasi* algoritmo.
Ebbene, per potere comunicare con l'automa, in modo che esso possa eseguire le istruzioni, dobbiamo codificarle in un certo modo, mediante un linguaggio che sia comprensibile sia a noi che all'automa stesso.
Un linguaggio di questo tipo viene detto ***linguaggio di programmazione***, e un algoritmo codificato mediante un linguaggio di programmazione viene detto ***programma***.

Per esempio, uno dei linguaggi di programmazione più diffusi è il *Pascal,* perché tra i linguaggi di alto livello, è fra quelli che meglio rispondono ai requisiti della programmazione strutturata. Di questo linguaggio esistono varie versioni, e fra tutte quella più diffusa è il *Turbo-Pascal*. Alcuni esempi di semplici programmi scritti in *Turbo-Pascal* sono i seguenti:

1] **Quoziente e resto di due numeri interi**

```
Program Quoziente_resto;
    var
        a, b: integer;
        q, r: byte;
   begin
        writeln('Introduci due numeri interi a e b);
        readln (a, b);
        q : = a div b;
        r : = a mod b;
        write ('Il quoziente e il resto della divisione a : b sono');
        writeln (q);
        writeln (r);
        readln;
end.
```

Già in questo semplice programma si possono evidenziare alcune caratteristiche del linguaggio Pascal facendo seguire un commento a ciascuna riga di istruzione.

```
Program Quoziente_resto;                    (Questa è l'intestazione del programma)
var                                         (iniziale della parola 'variabili')
a, b: integer; (si dichiara che a e b sono di tipo integer, cioè appartengono all'intervallo -32768 ... 32767)
q, r: byte;    (si dichiara che q ed r sono di tipo byte, cioè appartengono all'intervallo 0 ... 255)
begin                                       (è il comando di inizio del programma)
writeln ('Introduci due numeri interi a e b);   (significa: 'scrivi la frase e vai a capo')
readln (a, b);                  (significa che l'operatore deve introdurre i valori di a e b);
q : = a div b;                  (significa: assegna a q il quoziente della divisione intera di a per b);
r : = a mod b;                  (significa: assegna a r il resto della divisione intera di a per b);
write ('Il quoziente e il resto della divisione a : b sono');
writeln (q);
writeln (r);
readln;
end.
```

2] **Somma dei quadrati dei primi *n* numeri naturali**

```
program Somma_Quad;
var
I, N: shortint;             (I ed N appartengono all'intervallo -128 ... 127)
S, Quad: word;              (S e Quad appartengono all'intervallo 0 ... 65535)
begin
```

```
        I : = 0;                    (inizializzazione della variabile I)
        S : = 0;                    (inizializzazione della variabile S)
        writeln ('Questo programma calcola la somma dei quadrati dei primi N numeri
                naturali);
        writeln ('Quanto vale N?');
        readln (N);
                repeat              (questo è l'inizio della parte iterativa del programma)
                I : = I + 1;
                Quad : = I * I;
                S : = S + Quad;
                until I = N;
        writeln ('La somma richiesta è', S);
        readln;
end.
```

3] **Programma per calcolare i primi 20 numeri di Fibonacci**

Come sappiamo, i numeri di Fibonacci appartengono alla successione:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, . . .

e sono definiti in maniera ricorsiva dalle due formule

$$f_1 = f_2 = 1$$
$$f_{n+1} = f_n - f_{n-1} \ (n > 2)$$

Per costruire l'algoritmo abbiamo bisogno:
- di una variabile in cui memorizzare ogni volta il numero calcolato della successione, e che possiamo chiamare NFIB;
- di due variabili, per esempio A e B, in cui memorizzare i due termini precedenti;
- di una variabile che sia il 'contatore', che conti, cioè, quanti numeri sono stati generati e che ci indichi quando saremo arrivati a 20.

La variabile NFIB assumerà solo valori positivi e può diventare molto grande, così come le variabili A e B, per cui sarà meglio dichiararle del tipo word;
- la variabile I raggiungerà come valore massimo il 20, per cui sarà più conveniente dichiararla di tipo byte o shortint.

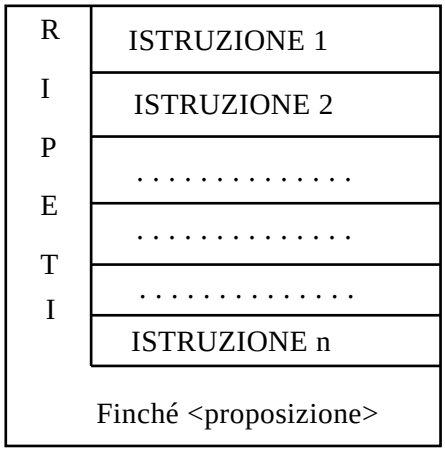
```
program FIBONACCI;
var
        NFIB, A, B : word;
        I: byte;
begin
        A : = 0;
        B : = 1;
        I : = 0;
        writeln ('I primi 20 numeri della successione di Fibonacci sono:');
                repeat
                NFIB : = A + B;
                writeln (NFIB);
                A : = B;
```

```
            B : = NFIB;
            I : = I + 1
            until I = 20;
      writeln (‘Premi INVIO per terminare l'esecuzione del programma’);
      readln;
end.
```

Notiamo che nel programma è presente la struttura dell'*iterazione per falso*

repeat ... until

il cui grafo di *Nassi-Schneidermann* è:



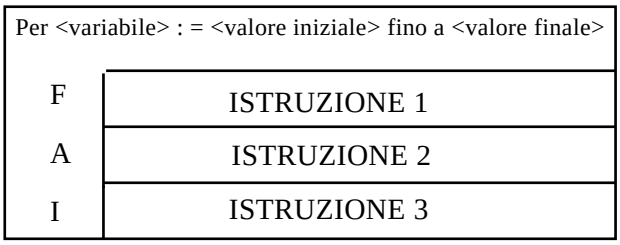
4] **Programma per calcolare la potenza *n*-esima di un numero A, eseguendo il prodotto di N fattori uguali ad A**.

```
program POTENZA;
      var
      N, I: byte;
      A, P: real;
        begin
            P : = 1; (inizializzazione della variabile potenza)
            writeln (‘Questo programma calcola la potenza n-esima di un numero A’);
            write(‘Introduci la base della potenza’);
            readln (A);
            write (‘Introduci il valore dell'esponente’);
            readln (N);
                  for I : = 1 to N do
                  P : = P * A;
            writeln (‘La potenza calcolata è, P:10:4);
        readln;
end.
```

Notiamo che in questo programma è presente la struttura dell'*iterazione enumerativa*

for ... to ... do

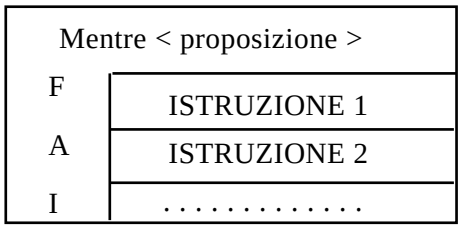
il cui grafo di *Nassi-Schneidermann* è:



5] Programma per calcolare la somma dei reciproci dei primi n numeri naturali

```
program SOMMA_INV;
var
        N, C: byte;
        S : real;
begin
        write (‘Di quanti numeri vuoi calcolare la somma dei reciproci?’);
        readln (N);
        S : = 0; (inizializzazione opzionale)
        C : = 1; (inizializzazione del contatore)
        while C < = N do
                begin
                        S : = S + 1 / C;
                        C : = C + 1;
                end;
        writeln (‘La somma richiesta è, S : 6 : 2) (Significa che il risultato è con 6 cifre di cui 2 decimali)
        readln;
end.
```

Si noti la presenza della struttura dell'*iterazione per vero*: **while ... do**, il cui grafo è:



Se supponiamo che il modo di operare del cervello umano, cosciente o no, sia semplicemente l'esecuzione di un qualche algoritmo molto complicato, dobbiamo chiederci in che modo un algoritmo così straordinariamente efficace abbia avuto origine. La risposta standard, ovviamente, sarebbe quella della selezione naturale. Nel caso di creature dal cervello evoluto, quelle con gli algoritmi più efficaci avrebbero avuto migliori probabilità di sopravvivenza e quindi, nel complesso, avrebbero lasciato una progenie più numerosa. Anche i loro discendenti avrebbero avuto la tendenza a presentare algoritmi più efficaci rispetto a quelli dei loro cugini, avendo ereditato gli ingredienti di questi algoritmi più efficaci dai loro genitori;

così gli algoritmi migliorarono gradualmente - non necessariamente in modo continuo, giacché la loro evoluzione potrebbe aver proceduto a salti - fino a raggiungere il livello notevole che noi riscontriamo oggi nel cervello umano.

Roger Penrose, *La mente nuova dell'Imperatore*, p. 523.

Esercitazioni

1) Descrivere le azioni da compiere per raggiungere i seguenti scopi:

a) vedere un programma televisivo;
b) attraversare la strada ad un incrocio con semaforo;
c) cuocere un uovo al tegamino.

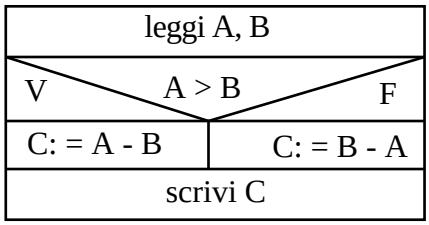
2) Dato l'algoritmo:

inizio
ricevi dall'esterno il numero 3
ricevi dall'esterno il numero 12
addizione i due numeri
fornisci il risultato
fine.

Quale problema risolve? Quali sono le competenze che deve avere un esecutore? Descrivere l'algoritmo a cui appartiene quello presentato.

3) Scrivere un algoritmo che, letto in ingresso un numero decimale *n*, dia in uscita il valore dell'espressione: $2 \cdot n \cdot (n + 5)$.

4) Dato il grafo di *Nassi-Schneidermann*



a) enunciare il problema che risolve;
b) tracciare il diagramma di flusso;
c) scrivere l'algoritmo in linguaggio di progetto.

5) Scrivere in LP (linguaggio di progetto) un algoritmo per determinare il successivo e il precedente di un numero intero.

6) Scrivere un algoritmo per determinare il quadrato e la radice quadrata di un numero con due variabili diverse.

7) Scrivere un algoritmo per determinare se un numero è divisibile per 2 o per 3.

8) Scrivere un algoritmo per determinare il perimetro e l'area di un rettangolo, note le sue dimensioni.

9) Scrivere un algoritmo per determinare la misura della diagonale di un rettangolo, note le sue dimensioni.

10) Scrivere un algoritmo per determinare l'area di un cerchio nota la misura del raggio.

11) Scrivere un algoritmo per determinare la misura della diagonale di un quadrato nota la sua area.

12) Scrivere un algoritmo per determinare il volume di un oggetto, noti il peso e il peso specifico.

13) Scrivere un algoritmo per determinare il volume di un cilindro, noti il raggio di base e la sua altezza.

14) Scrivere un algoritmo per determinare il peso di un oggetto a forma di cono, noti il raggio di base, il suo apotema e il peso specifico del materiale di cui è fatto.

15) Scrivere un algoritmo per determinare il volume di un cubo, il cui spigolo è uguale a quello di un altro cubo aumentato di un valore *x*.

16) Scrivere un algoritmo per determinare la radice quadrata di un numero.

17) Scrivere un algoritmo per determinare le soluzioni di un'equazione di primo grado.

18) Scrivere un algoritmo per determinare se un numero è multiplo di un altro numero.

19) Scrivere un algoritmo per generare le successioni:

1, 3, 5, 7, 9, ... , 99.
1, 4, 9, 16, 25, ... , 2500.
1, - 2, 3, - 4, 5, - 6, ... , - 100.
$\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \ldots, \frac{99}{100}$

20) Scrivere un algoritmo per leggere 20 numeri e stabilire qual è il maggiore tra essi.

21) Scrivere un algoritmo per determinare la somma dei primi 100 numeri naturali.

22) Scrivere un algoritmo per determinare la somma dei primi N numeri naturali.

23) Scrivere un algoritmo per determinare il prezzo di un oggetto scontato di un valore percentuale *x*.

24) Scrivere un algoritmo per determinare la somma di tre numeri consecutivi.

25) Scrivere un algoritmo per determinare l'area di un cerchio conoscendo la sua circonferenza.

26) Scrivere un algoritmo per convertire in radianti la misura di un angolo dato in gradi/primi/secondi.

27) Scrivere un algoritmo per convertire in gradi/primi/secondi la misura di un angolo dato in radianti.

28) Scrivere un algoritmo per convertire un numero scritto in base 10 in un altro scritto in base 2.

29) Scrivere un algoritmo per convertire un numero scritto in base 2 in un altro scritto in base 10.

30) Scrivere un algoritmo per determinare il m.c.m. fra due numeri usando la formula

$$\text{m.c.m. (A, B)} = \frac{A \cdot B}{\text{M.C.D. (A, B)}}$$

31) Scrivere un algoritmo per determinare se due numeri sono primi fra loro.

32) Scrivere un algoritmo per determinare tutte le terne di numeri *a*, *b*, *c*, con *a* e *b* compresi tra 1 e 100, tali che $a^2 + b^2 = c^2$.

33) Scrivere un algoritmo per determinare i primi 20 termini di una progressione aritmetica, conoscendo il primo termine e la ragione.

34) Scrivere un algoritmo per determinare i primi 20 termini di una progressione geometrica, conoscendo il primo termine e la ragione.

35) Scrivere un algoritmo per risolvere un sistema di due equazioni lineari in due incognite con il metodo di Cramer.

36) Scrivere un algoritmo che determini la media quadratica di *N* numeri.

37) Scrivere un algoritmo che determini la media armonica di *N* numeri.

38) Nel gioco degli *11* fiammiferi due giocatori, a turno, devono prelevare un numero di fiammiferi compreso tra *1* e *3*. Vince il giocatore che costringe l'avversario a prendere l'ultimo fiammifero. Per chi apre il gioco esiste una strategia vincente: deve prelevare *2* fiammiferi alla prima mossa, e *4 - n* fiammiferi nelle mosse successive, dove *n* è il numero di fiammiferi presi dall'avversario. Costruire un algoritmo che stabilisca ad ogni mossa il numero di fiammiferi rimasti e la strategia vincente per il primo giocatore.

39) Scrivere un algoritmo tale che, se un numero è minore o uguale a 50 ne faccia il quadrato, altrimenti scriva il suo doppio.

40) Scrivere un algoritmo che calcoli il più piccolo intero positivo il cui quadrato è maggiore di un numero positivo *n*.

L'algoritmo euclideo

Gli antichi furono maestri nell'arte di definire algoritmi e, più in generale, di risolvere problemi mediante tecniche costruttive.

Paolo Zellini, Gnomon, *Adelphi, 1999, p. 55.*

Anche se esistono serie obiezioni all'ipotesi - avanzata in diverse occasioni - che la scoperta di grandezze incommensurabili sia avvenuta con l'algoritmo delle sottrazioni successive, è legittimo sospettare che tale algoritmo fosse conosciuto in tempi molto antichi e che fosse, prima di Euclide, il fondamento stesso dell'idea di rapporto.

W.Knorr, Aristotle and Incommensurability, *1981.*

a) ***Il contesto storico***

L'algoritmo euclideo per calcolare il massimo comun divisore tra due o più numeri viene esposto da Euclide (circa *300* a.C.) nelle prime tre proposizioni del *Libro VII* dei suoi *Elementi*.

Euclide

Prima di Euclide, non vi è alcun riferimento esplicito a questo metodo, noto appunto come '*algoritmo euclideo*', anche se alcune tracce si trovano in opere precedenti agli *Elementi*. Così, per esempio, Aristarco di Samo (circa III sec. a.C.), nell'opera *Sulle dimensioni e le distanze del Sole e della Luna*, sostituisce il rapporto $\frac{71.755875}{61.735500}$ con l'altro $\frac{43}{37}$, e usa il rapporto $\frac{88}{45}$ al posto del rapporto $\frac{7921}{4050}$.

In un'epoca posteriore a quella d'Euclide, Archimede di Siracusa (287-212 a.C.), nell'opera *Misura del cerchio*, rimpiazza il rapporto

$$\frac{6336}{2017} + \frac{1}{4}$$

con il numero

$$3 + \frac{10}{71} \left(= \frac{223}{71} \right).$$

È chiaro che queste tre approssimazioni furono ottenute usando l'algoritmo euclideo che, per evitare anacronismi e per rimanere quanto più fedeli al testo originale degli *Elementi*, viene anche detto a volte *antanairesis* (o *antifairesis*) o *sottrazione alternata*.

Nella prima proposizione, viene trattato il caso in cui i numeri siano primi tra loro, aventi cioè come massimo comun divisore l'unità; nella seconda proposizione si considera il caso di due numeri che non sono primi tra loro, e nella terza quello di tre numeri, pure non primi tra loro.

Lo stesso algoritmo verrà usato da Euclide nel *Libro X* degli *Elementi*, nelle proposizioni 2^a, 3^a e 4^a per determinare la massima comune misura tra due o tre grandezze (prop. 3^a e 4^a).

Nel caso dei numeri, il procedimento ha sempre termine, mentre, nel caso di grandezze, il procedimento avrà termine solo se le grandezze saranno *commensurabili*; altrimenti, quando sono *incommensurabili*, esso non avrà termine.
Per ben comprendere le proposizioni euclidee, si deve tener presente che per Euclide il *numero* (αριθμοσ) è ciò che noi chiamiamo *numero naturale* o equivalentemente *numero intero positivo.* Invece, la *grandezza* poteva essere una qualsiasi quantità finita continua, come un segmento di retta, una figura piana limitata da segmenti di retta, una figura solida limitata da figure piane. I numeri potevano essere interpretati come particolari grandezze, ovvero erano *misure* di grandezze, e in generale Euclide li rappresenta sempre come segmenti di retta.
Per Euclide il misurare consisteva nello stabilire quante volte un numero era contenuto in un altro numero, e il risultato di una misura stabiliva il numero di unità che componevano un certo numero intero.
Se *m* ed *n* erano due numeri, ed *n* era contenuto in *m* un numero esatto di volte, per esempio *k* volte, allora si diceva che *m* misurava esattamente *k*.
L'operazione che stava alla basa di questa tecnica del confronto tra numeri, cioè del *misurare*, era l'*antanairesis*, ovvero quella del *sottrarre*, del *togliere*; per cui, misurare un numero *b* con un numero minore, *a*, significava togliere *a* da *b* quante volte era possibile, rimanendo eventualmente con un resto, *r*, che risultava più piccolo di *a*. Nel caso più semplice, quando il resto *r* era nullo, allora *b* era divisibile *esattamente* per *a*, e in tal caso *a* e *b* ammettevano una *massima misura comune*. Se, invece, *r* non era nullo allora esso si toglieva da *a* quante volte era possibile, ottenendo così un nuovo resto *r'*, più piccolo di *r*, che era nullo nel caso in cui *r* divideva esattamente *a*, e in tal caso *r'* era quello che si chiamava il «massimo comun divisore» di *a* e *b*. Se, invece, *r'* non era nullo, l'algoritmo procedeva finché non si otteneva un resto nullo; inoltre, esso aveva sicuramente termine, perché i resti erano numeri interi positivi e diventavano sempre più piccoli. L'ultimo resto non nullo risultava essere il massimo comun divisore tra *a* e *b*, e se questo resto era *1*, allora *a* e *b* erano *primi tra loro*.
Si può quindi dire che l'algoritmo euclideo era un modo per risolvere la questione della divisione tra due interi positivi *a* e *b*; ovvero di scoprire se *b* era esattamente divisibile per *a* oppure no; cioè, come dice Euclide, per scoprire se *a* era *parti* di *b*, circostanza, quest'ultima, che si traduceva nella possibilità di scomporre sia *a* che *b* in una somma di addendi tutti uguali al loro massimo comun divisore *d*:

$$a = d + d + d + d + \ldots + d \quad (m \text{ volte})$$

$$b = d + d + d + d + \ldots + d \quad (n \text{ volte})$$

Cosa diversa era, invece, dire che un numero *a* era *parte* di un secondo numero *b*, perché, in tal caso, ciò significava che *a* era un sottomultiplo di *b*. Così, per esempio, *2* è *parte* di *6*, perché lo divide; mentre *4* è *parti* di *6*, perché non lo divide.

Una pagina degli Elementi *di Euclide*

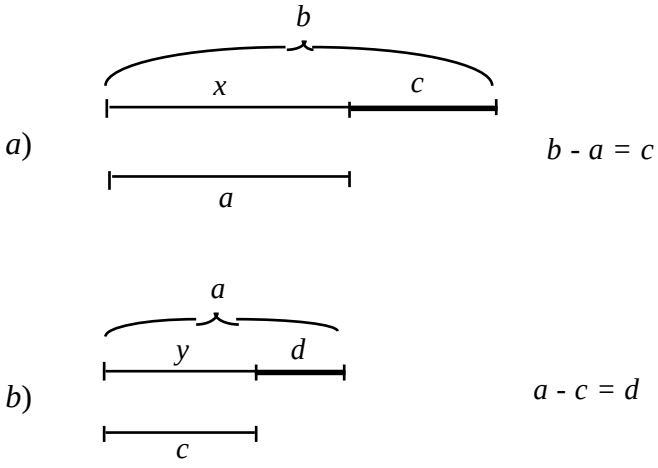
b) Le proposizioni fondamentali del Libro VII

Proposizione 2

Dati due numeri che non siano primi tra loro, trovare il loro massimo comun divisore (in greco: «la loro massima misura comune»).

Dimostrazione della proposizione

Euclide considera innanzitutto il caso in cui *a* divida esattamente *b* (ovvero è contenuto un numero intero di volte in *b*); ma *a* divide anche se stesso, quindi *a* è *divisore comune* di *a* e di *b*; ed è anche il *massimo*, perché nessun numero maggiore di *a* può dividere *a* stesso.
Si supponga, ora, che *a* non divida esattamente *b*, e si cominci a sottrarre il numero minore dal maggiore, la differenza ottenuta dal minore, e così via. Si otterrà, allora, un numero che dividerà esattamente il numero immediatamente precedente. Rappresentando, come fa Euclide, i due numeri *a* e *b* come misure delle lunghezze di due segmenti, si può visualizzare il procedimento nel modo che segue:



Si supponga, dice Euclide, che *d* divida *c*; ma siccome *c* divide *y*, allora *d* dividerà *y* + *d* = *a*; ma *a* divide *x*, per cui *d* dividerà pure *x* + *c* = *b*; per cui *d* è *divisore comune* di *a* e di *b*. Il numero

d è anche il *massimo*. Supponiamo, infatti, che non lo sia. Allora vorrà dire che vi sarà un numero *d' maggiore* di *d* che divide sia *a* che *b*. Quindi, poiché *d'* divide *a*, ed *a* divide *x*, *d'* dividerà *x*; ma poiché esso divide *b*, dividendo *x* dividerà pure *c*. Ma poiché *c* divide *y*, anche *d'* dividerà *y*; ma, dividendo *a*, allora esso dividerà pure *d*. Ciò è assurdo, perché *d'* dividerebbe un numero minore di esso. Si conclude che il numero *d* è proprio l'unico *massimo comun divisore* di *a* e *b*.■

Il ragionamento euclideo di questa seconda parte della dimostrazione si può schematizzare nel modo seguente:

IPOTESI: esiste un numero $d' > d$ tale che $d' \mid a$ e $d' \mid b$.
(L'espressione $d' \mid a$ si legge: «*d'* divide *a*»)

Poiché $d' \mid a$
ed $a \mid x$
allora: $d' \mid x$.

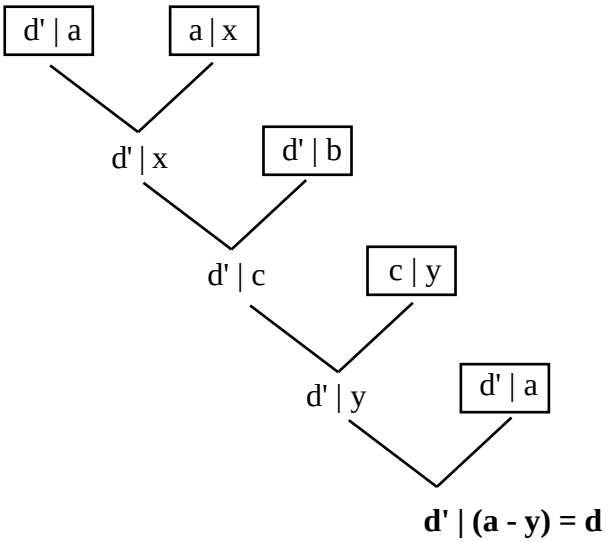
Se $d' \mid x$
e $d' \mid b$
allora: $d' \mid (b - x) = c$.

Se $d' \mid c$
e $c \mid y$
allora: $d' \mid y$.

Se $d' \mid y$
e $d' \mid a$
allora: $\boldsymbol{d' \mid (a - y) = d}$.

Assurdo! perché $\boldsymbol{d' \mid d}$ *contro l'ipotesi* che $\boldsymbol{d' > d}$.

La stessa dimostrazione può rappresentarsi con un diagramma ad *albero*:



Importante è il *corollario* di questa proposizione:

Se un numero c divide altri due numeri a e b, allora divide anche il loro massimo comun divisore d.

Infatti, se $a > b$, supponendo, ad esempio, che l'algoritmo euclideo abbia termine dopo quattro passi, potremmo scrivere:

$$a = m\,b + r$$
$$b = n\,r + r'$$
$$r = p\,r' + r''$$
$$r'' = q\,r'''$$

quindi $r''' = d$, cioè il massimo comun divisore di a e b.
Ebbene, se un numero c dividesse a e b, allora si avrebbe:

$$a = k\,c$$
$$b = h\,c$$

e la prima delle relazioni precedenti ci darebbe:

$$c \cdot (k - hm) = r$$

cioè, c dividerebbe r; dalla seconda relazione si avrebbe:

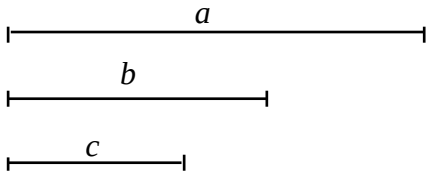
$$c \cdot [h - n\,(k - hm)] = r'$$

cioè, c dividerebbe r'; similmente, c dividerebbe r'' ed $r''' = d$.

Proposizione 3

Dati tre numeri che non siano primi fra loro, trovare il loro massimo comun divisore.

Come per la proposizione *2*, anche per questa Euclide considera tre segmenti di lunghezza rispettiva *a*, *b* e *c*.



Sia *d* il massimo comun divisore di *a* e di *b*. Sono allora possibili due casi:
i) $d \mid c$, allora esso è divisore di *a*, *b* e *c*. Esso è pure il massimo. Supponiamo, infatti, che *d* non lo sia, per cui esisterebbe un numero $e > d$ che dividerebbe *a*, *b* e *c*. Ma allora *e* dividerebbe anche il massimo comun divisore di *a* e *b*, cioè *d*; il che è assurdo. Il numero *d* è quindi il massimo comun divisore di *a*, *b* e *c*.
ii) $\neg\,(d \mid c)$ [cioè, *d* non divide *c*]. Ebbene, *d* e *c* non sono primi tra loro. Infatti, per ipotesi, *a*, *b* e *c* non sono primi fra loro, per cui li dividerà un qualche numero. Poiché questo numero divide *a* e *b*, dividerà anche il loro massimo comun divisore, cioè *d*. Ma tale numero divide pure *c*, per cui sarà divisore sia di *d* che di *c*, per cui *c* e *d* non saranno primi fra loro. Sia dunque *e* il massimo comun divisore di *c* e *d*. Poiché *e* divide *d*, e *d* divide *a* e *b*, anche *e* dividerà *a* e *b*; ma *e* divide pure *c*, per cui esso sarà divisore comune di *a*, *b* e *c*. Ebbene, esso è anche il massimo. Infatti, si supponga che *e* non sia il massimo comun divisore di *a*, *b* e *c*; allora esisterà un numero $f > e$ che dividerebbe *a*, *b* e *c*. Quindi, poiché *f* divide *a* e *b*, allora dividerebbe anche il loro massimo comun divisore, cioè *d*. Ma *f* divide pure *c*, per cui divide *c*, *d* e anche il loro massimo comun divisore, cioè *e*; ciò è *chiaramente assurdo*, perché si è supposto $f > e$. ■

Anche nella dimostrazione di questa proposizione si nota come Euclide applica la stessa procedura dimostrativa usata per dimostrare la proposizione *2*, per cui si potrebbe dire che per dimostrare la validità di un *algoritmo computazionale* Euclide usa un *algoritmo dimostrativo*.
L'algoritmo euclideo serve oggi per caratterizzare proprio il cosiddetto *anello euclideo*, ovvero un anello *A* che possiede un 'algoritmo euclideo', cioè un'applicazione *g* di *A* in N tale che: dati due elementi *a* e *b* di *A*, con *b* non nullo, esistono due elementi *q* ed *r* di *A* che soddisfano la relazione:

$$a = bq + r \qquad \text{con } gr(r) < gr(b).$$

c) *La ricerca del massimo comun divisore nella didattica odierna*

L'algoritmo euclideo procede, quindi, come s'è visto, mediante sottrazioni successive, e per un ragazzo sottrarre un numero da un altro riesce più «facile» che dividere il primo per il secondo. Didatticamente, anche se a volte può risultare molto lungo, esso è agevole, ma, *inspiegabilmente*, al metodo delle sottrazioni successive i libri di testo scolastici preferiscono adottare un altro procedimento per calcolare il massimo comun divisore tra due numeri, basato proprio sul concetto di fattorizzazione di un numero composto. Ma, com'è noto, la fattorizzazione di un numero composto spesso non è tanto semplice, per cui si genera nel discente la sensazione che la ricerca del massimo comun divisore tra i numeri sia una delle *operazioni* matematiche da evitare ... accuratamente.
Consideriamo, a questo proposito, un esempio numerico. Si voglia determinare il massimo comun divisore tra *5568* e *864*; esso verrà indicato con la scrittura (*5568, 864*).
Le istruzioni di calcolo che vengono impartite all'allievo sono generalmente le seguenti:
a) Fattorizza *5568* e *864*:

5568	2
2784	2
1392	2
696	2
348	2
174	2
87	3
29	29
1	

864	2
432	2
216	2
108	2
54	2
27	3
9	3
3	3
1	

b) Scrivi *5568* e *864* mediante il prodotto delle potenze dei suoi fattori primi:

$$5568 = 2^6 \cdot 3 \cdot 29$$

$$864 = 2^5 \cdot 3^3$$

c) Tra le potenze aventi uguale base scegli quelle con l'esponente minore:

$$2^5 , 3$$

d) Il massimo comun divisore tra *5568* e *864* sarà allora uguale al prodotto $2^5 \cdot 3$:

$$(5568, 864) = 2^5 \cdot 3 = 96.$$

Dopo di che viene dedotta la famosa regola:

Il massimo comun divisore tra due o più numeri è uguale al prodotto dei fattori primi comuni presi una sola volta con il minimo esponente.

Affinché lo studente si renda conto del «perché» debba prendere come massimo comun divisore proprio il prodotto delle potenze di quei due numeri, di solito si scrivono i divisori successivi sia di *5568* che di *864* e gli si fa notare che il loro massimo comun divisore coincide con il prodotto $2^5 \cdot 3$, secondo lo schema seguente:

	DIVISORI															
5568	2	3	4	6	8	12	16	24	29	32	64	87	**96**	116	. . .	
864	2	3	4	6	8	9	12	18	24	27	32	36	54	72	**96**	144 . .

Ora, in questo caso, non si sono dovuti scrivere tutti i divisori dei due numeri, perché il massimo comune divisore si è *incontrato* abbastanza presto. Ma, spesso, bisogna scrivere tutti o quasi tutti i divisori dei numeri proposti, prima di verificare qual è il *divisore più grande che hanno in comune.* E la difficoltà consiste proprio nel fatto che, a volte, i divisori di un numero sono proprio molti. Si dimostra, infatti, che se un numero *n* si può fattorizzare nella forma:

$$n = p_1^{\alpha_1} \; p_2^{\alpha_2} \; p_3^{\alpha_3} \; \ldots \; p_r^{\alpha_r}$$

dove con p_i si è indicato il generico fattore primo e con α_i la molteplicità di p_i, allora il numero dei divisori di *n* è dato dalla formula:

$$d(n) = (\alpha_1 + 1)\,(\alpha_2 + 1) \ldots (\alpha_r + 1)$$

Per cui, tornando all'esempio di prima, poiché

$$5568 = 2^6 \cdot 3^1 \cdot 29^1$$

$$864 = 2^5 \cdot 3^3$$

i divisori di *5568* sarebbero $(6 + 1)\,(1 + 1)\,(1 + 1) = 28$, e quelli di *864*, $(5 + 1)\,(3 + 1) = 24$.
Tutto il procedimento ha comunque l'unica valenza didattica di fare esercitare l'allievo sulla fattorizzazione dei numeri, mentre si perde la connotazione precipua della determinazione del massimo comun divisore, ripescata solo alla fine, magari dopo avere eseguito delle fattorizzazioni estenuanti.
L'algoritmo euclideo originale non perde, al contrario, durante la sua esecuzione, la connotazione che lo contraddistingue proprio per lo scopo per cui è stato inventato, ovvero per la ricerca del massimo comun divisore. Infatti, lo studente, a mano a mano che procede nel «calcolo» sa che le successive uguaglianze che scrive lo condurranno, dopo una serie di passi, alla determinazione di ciò che cerca.
L'algoritmo euclideo originale può allora essere presentato e giustificato agli studenti utilizzando il nostro simbolismo. Basta fargli osservare quanto segue. Se il massimo comun divisore tra i numeri *a* e *b* è il numero *c*, allora possiamo scrivere sia *a* che *b* come prodotto di *c* per un numero naturale opportuno:

$$a = c \cdot k$$
$$b = c \cdot h$$

Se *a* e *b* sono diversi, allora anche *h* e *k* saranno diversi; altrimenti la determinazione del loro massimo comun divisore sarebbe immediata perché il massimo comun divisore di un numero e se stesso coincide con il numero stesso. Supponiamo dunque che sia $a > b$. Potremo allora scrivere:

$$a - b = c \cdot k - c \cdot h = c \cdot (k - h)$$

quindi *c* dividerebbe pure la differenza $(a - b)$. Quindi il numero *c* è anche il massimo comun divisore di *b* e di $a - b$. Infatti, *c* li divide entrambi; inoltre, se esso non fosse il massimo dovrebbe esiste un divisore comune $d > c$ che ci permetterebbe di scrivere:

$$a - b = d \cdot m \qquad \text{da cui: } a = d \cdot m + b$$

$$b = d \cdot n$$

per cui si avrebbe:

$$a = d \cdot m + d \cdot n = d \cdot (m + n)$$

Quindi, *d* sarebbe pure massimo comun divisore tra *a* e *b* e dovrebbe allora dividere anche *c*, il che è assurdo, avendo supposto $d > c$.■

Basta questo per comprendere che se dobbiamo calcolare il massimo comun divisore tra due numeri *a* e *b* ($a > b$), si può calcolare quello tra due numeri più piccoli: $(a - b)$ e *b*, e che questo ragionamento si può ripetere calcolando il massimo comun divisore tra la loro differenza e il più piccolo tra i due, finché si arriva a calcolare il massimo comun divisore tra due numeri uguali: *il loro valore sarà allora il massimo comun divisore cercato tra i numeri dati*.
Calcoliamo, per esempio, il massimo comun divisore tra $a = 480$ e $b = 225$.

$(480, 225) =$
$(255, 225) =$ [sottraiamo: $480 - 225 = 255$ e lo mettiamo al posto di *480*]
$(225, 30) =$ [sottraiamo: $255 - 225 = 30$ e lo mettiamo al posto di *255*]
$(195, 30) =$ [sottraiamo: $225 - 30 = 195$ e lo mettiamo al posto di *225*]
$(165, 30) =$ [sottraiamo: $195 - 30 = 165$ e lo mettiamo al posto di *195*]
$(135, 30) =$ [sottraiamo: $165 - 30 = 135$ e lo mettiamo al posto di *165*]
$(105, 30) =$ [sottraiamo: $135 - 30 = 105$ e lo mettiamo al posto di *135*]
$(75, 30) =$ [sottraiamo: $105 - 30 = 75$ e lo mettiamo al posto di *10*]
$(45, 30) =$ [sottraiamo: $75 - 30 = 45$ e lo mettiamo al posto di *75*]
$(30, 15) =$ ***15*** [sottraiamo: $45 - 30 = 15$ e lo mettiamo al posto di *45*]

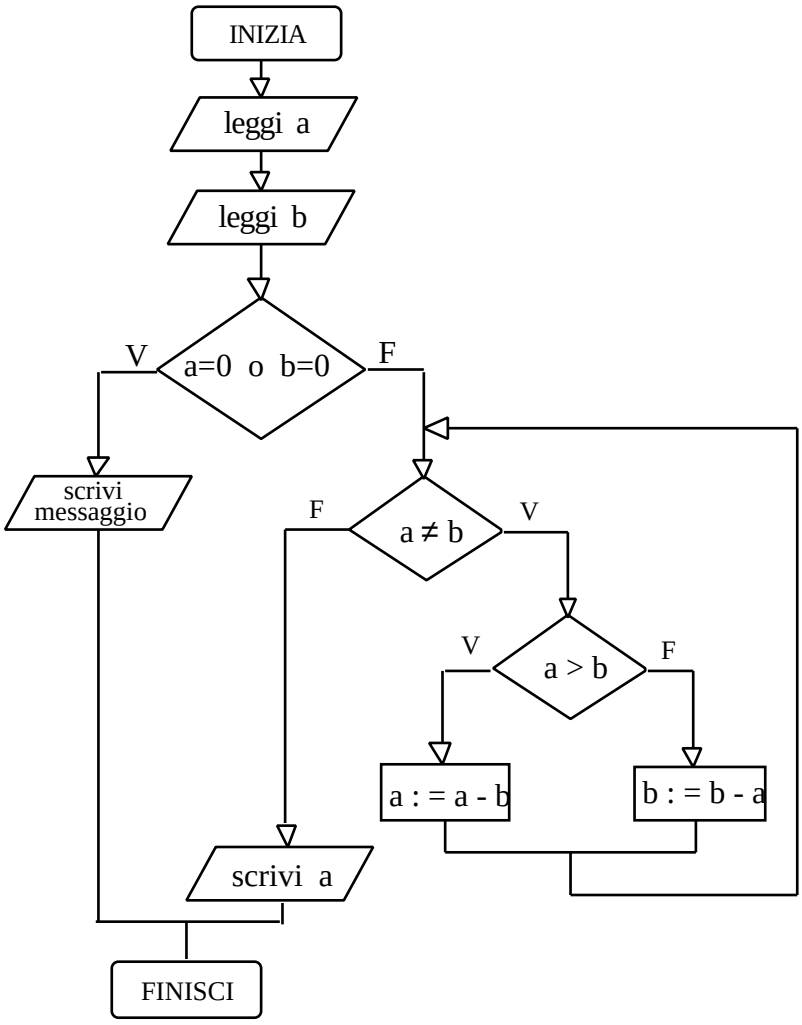
Lo stesso procedimento è equivalente a successive divisioni, perché possiamo scrivere:

$480 = 2 \cdot 225 + 30$ [dividendo *480* per *225*, otteniamo *2* come quoto e *30* come resto]

$225 = 7 \cdot 30 + 15$ [dividendo *225* per *30*, otteniamo *7* come quoto e *15* come resto]

$30 = 2 \cdot \mathbf{15} + 0$ [dividendo *30* per *15*, otteniamo *2* come quoto e *0* come resto]

Il diagramma di flusso dell'algoritmo e la sua traduzione in linguaggio di progetto saranno:




```
Variabili a, b: naturali;
inizia
    leggi a;
    leggi b;
    se a = 0 o b = 0 scrivi 'errore!'
    altrimenti inizia
                    mentre a ≠ b fai
                        inizia
                            se a > b allora a : = a - b;
                            altrimenti b : = b - a
                        finisci;
                    scrivi a
                    finisci
finisci.
```

Se in N introduciamo una nuova operazione tale che a due numeri naturali venga associato il resto della loro *divisione intera,* potremo rappresentare l'algoritmo euclideo con un altro diagramma di flusso. Indichiamo questa operazione con *mod* e definiamola così: se *a* e *b* sono due numeri naturali, con *a* > *b*, allora:

a mod b = r dove *r* è il resto della divisione di *a* per *b*.

Quindi $r = a - bq$, dove q è il quoziente della divisione. Ora, se d è il massimo comun divisore di a e b, allora essi possono scriversi come

$$a = k \cdot d$$

$$b = h \cdot d$$

per cui si ha:

$$r = k \cdot d - h \cdot d\, q$$

$$= d \cdot (k - h\, q).$$

Il numero d è dunque divisore di r. Allora, per calcolare il massimo comun divisore di a e b, calcoliamo il massimo comun divisore di b e di $a \bmod b$:

$$MCD\ (a, b) = MCD\ (b, a \bmod b).$$

Procedendo in questo modo, un numero diventerà zero, e l'altro sarà il massimo comun divisore dei numeri dati.

Seguiamo un esempio:

(*495, 60*) =

= (*60, 15*) [*485* diviso per *60* dà *8* come quoto e *15* come resto]

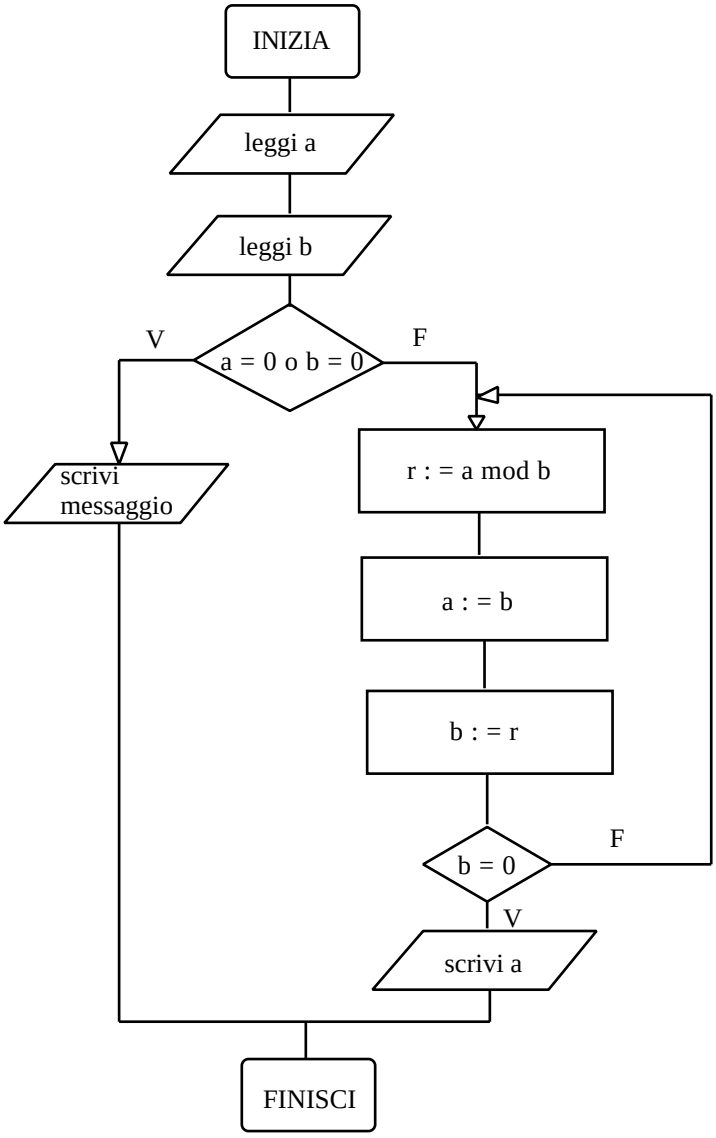
= (*15, 0*) [*60* diviso per *15* dà *4* come quoto e *0* come resto].

Il massimo comun divisore di *495* e *60* è quindi *15*.

L'algoritmo generale potrà essere rappresentato, in linguaggio di progetto, nel modo seguente:

```
variabili a, b, r: naturali;
inizia
        leggi a;
        leggi b;
        se a = 0 o b = 0 scrivi 'errore!'
        altrimenti inizia
                        ripeti  r : = a mod b
                                a : = b;
                                b : = r
                        finché b = 0;
                finisci
finisci.
```

Il corrispondente diagramma di flusso sarà:



Come si nota, la forma delle divisioni successive dell'algoritmo euclideo è più efficace della forma delle sottrazioni successive, perché è necessario un numero minore di passi per determinarlo. A questo proposito, il matematico francese Gabriel Lamé (1795-1871) dimostrò che *il numero di divisioni necessarie per determinare il massimo comun divisore di due numeri è al più cinque volte il numero delle cifre del numero più piccolo.*
Così, per esempio, se dobbiamo determinare il massimo comun divisore tra *1804* e *328*, con la divisione ordinaria si trova che:

$$\begin{array}{r|l} 1804 & 328 \\ \hline 164 & 5 \end{array}$$

per cui

$$1804 = 5 \cdot 328 + 164$$

Una seconda divisione dà:

$$\begin{array}{r|l} 328 & 164 \\ \hline 0 & 2 \end{array}$$

Quindi abbiamo determinato il massimo comun divisore tra *1804* e *328*, che è *164*, solo con due divisioni.
Lo stesso algoritmo, utilizzato nella forma delle sottrazioni successive avrebbe comportato i successivi passaggi:

$$\begin{aligned} (1804, 328) &= \\ &= (1476, 328) \\ &= (1148, 328) \\ &= (820, 328) \\ &= (328, 164) \\ &= (164, 164) = 164. \end{aligned}$$

d) ***Divisione e resti***

Come s'è visto, *se a e b (b > 0) sono due interi, si può sempre trovare un numero intero q tale che*

$$a = b\,q + r$$

dove r è un numero intero che soddisfa alla disuguaglianza $0 \le r < b$, *ovvero, r è uno dei numeri:*

$$0, 1, 2, 3, \ldots, b - 1.$$

Il resto *r* viene detto, in questo caso, il *minimo resto positivo* della divisione di *a* per *b*.
Questo fatto si può dimostrare in modo semplice. Infatti, ogni intero *a* è o multiplo di *b*,

$$a = b\,q$$

oppure è compreso tra due multipli successivi di *b*,

$$b\,q < a < b\,(q + 1) = b\,q + b.$$

Nel caso che

$$a = b\,q$$

l'uguaglianza

$$a = b\,q + r$$

è soddisfatta per $r = 0$.
Nel caso che

$$b\,q < a < b\,(q + 1) = b\,q + b.$$

sottraendo *b q* da ogni termine si ha:

$$b\,q - b\,q < a - b\,q < b\,q + b - b\,q$$

cioè

$$0 < a - b\,q < b$$

ovvero $0 < r < b$ com'è richiesto. ■
Ebbene, la stessa uguaglianza *a = b q + r si può scrivere nella forma:*

$$\frac{a}{b} = q + \frac{r}{b}$$

dove $\frac{r}{b}$ è zero oppure una frazione positiva minore di *1*, e *q* è il più grande intero positivo minore o uguale ad $\frac{a}{b}$. Quest'ultimo fatto si può rappresentare nella forma

$$q = \left[\frac{a}{b}\right]$$

che rappresenta appunto *il più grande intero contenuto in* $\frac{a}{b}$.

Per esempio:

$$\left[\frac{27}{5}\right] = 5, \qquad \left[\frac{5}{3}\right] = 1, \qquad [2] = 2, \qquad \left[\frac{-1}{3}\right] = -1, \qquad \left[\frac{1}{3}\right] = 0$$

A volte è più vantaggioso rappresentare la divisione tra due numeri *a* e *b* in una maniera leggermente differente dalla forma

$$a = b\,q + r$$

Si considera, cioè, un multiplo *k b* il più vicino possibile ad *a* e si scrive

$$a = k\,b + s$$

dove *s* è un numero compreso tra $-\frac{b}{2}$ e $\frac{b}{2}$.

In questo caso, *s* si chiama il *minimo resto assoluto* della divisione di *a* per *b*. Così, se consideriamo, per esempio, i numeri *35* e *9*, la loro divisione con il *minimo resto positivo* sarà rappresentata da

$$35 = 3 \cdot 9 + 8$$

mentre la loro divisione con il *minimo resto assoluto* sarà rappresentata da

$$35 = 4 \cdot 9 - 1.$$

Ora, nel caso della divisione ordinaria, il resto è sempre univocamente determinato. Questo succede anche nel caso della divisione con il minimo resto assoluto, tranne nel caso in cui *b* sia pari ed il resto sia $s = \pm\frac{b}{2}$, per cui *a* può essere rappresentato in due modi:

$$a = k\,b + \frac{b}{2}$$

$$a = (k + 1)\,b - \frac{b}{2}$$

Ma anche in questo caso, se si desidera avere sempre un unico resto si può scegliere come resto $s = \frac{b}{2}$.

Se consideriamo, per esempio, $a = 45$ e $b = 6$, si avrebbero le due rappresentazioni:

$$45 = 7 \cdot 6 + 3$$

$$45 = 8 \cdot 6 - 3$$

In questo caso, per avere un resto unico si può scegliere la prima rappresentazione, come detto precedentemente.
Qual è l'utilità dei minimi resti assoluti quando si divide un numero per un altro? La risposta alla domanda viene da un risultato dovuto al matematico tedesco Leopold Kronecker (1823-1891), uno dei maggiori studiosi di teoria dei numeri del secolo diciannovesimo, che dimostrò che *nessun algoritmo euclideo può essere più breve di quello ottenuto per mezzo dei minimi resti assoluti.*
Mettiamo a confronto, per esempio, l'algoritmo euclideo solito con quello ottenuto con i minimi resti assoluti per due numeri, $a = 76.084$ e $b = 63.020$.

L'algoritmo euclideo delle divisioni con i minimi resti positivi è il seguente:

$$76.084 = 63.020 \cdot 1 + 13.064$$

$$63.020 = 13.064 \cdot 4 + 10.764$$

$$13.064 = 10.764 \cdot 1 + 2300$$

$$10.764 = 2300 \cdot 4 + 1564$$

$$2300 = 1564 \cdot 1 + 736$$

$$1564 = 736 \cdot 2 + 92$$

$$736 = 92 \cdot 8$$

Richiede, quindi, *sette* passi.

L'algoritmo euclideo delle divisioni con i minimi resti assoluti è invece il seguente:

$$76.084 = 63.020 \cdot 1 + 13.064$$

$$63.020 = 13.064 \cdot 5 - 2300$$

$$13.064 = 2300 \cdot 6 - 736$$

$$2300 = 736 \cdot 3 + 92$$

$$736 = 92 \cdot 8$$

Esso ha richiesto *cinque* passi.
Quindi, didatticamente, offrire al discente i due modi di rappresentare la divisione tra due numeri significa fargli verificare l'efficacia del secondo rispetto al primo, perché è migliorato il *tempo di esecuzione*. Per un algoritmo, ciò ha molta importanza, perché, a parità di risultati, si sceglie sempre l'algoritmo che raggiunge lo scopo in un tempo minore.

e) Quattro variazioni didattiche

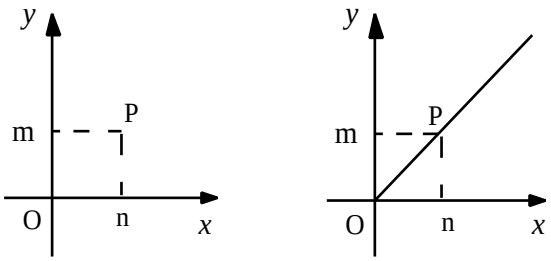
Per rendere più intuitiva la comprensione del massimo comun divisore tra due numeri, lo si può visualizzare[1] efficacemente nel modo che segue.

[1] Cfr. G.Cesare Barozzi, *L'algoritmo euclideo per il calcolo del massimo comune divisore: quattro variazioni facili su tema classico*, Archimede, 1986, pp. 79-90.

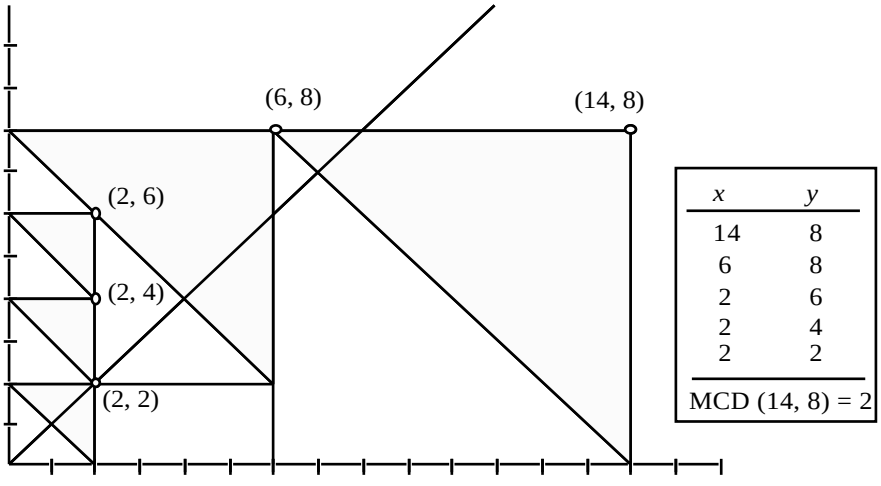
Per rendere più intuitiva la comprensione del massimo comun divisore tra due numeri, lo si può visualizzare efficacemente nel modo che segue.

I variazione.

Siano *n* ed *m* due interi positivi. La coppia (*n*, *m*) si può rappresentare mediante un punto del piano cartesiano.



Se *n* = *m*, allora il punto sta sulla diagonale del primo quadrante, cioè appartiene alla retta di equazione *y*=*x*, per cui il valore comune delle sue coordinate fornisce il MCD (massimo comun divisore) cercato.
Se *n* ≠ *m* si consideri il più piccolo triangolo rettangolo isoscele, avente l'angolo retto nel punto (*n*, *m*), i cateti paralleli agli assi coordinati, ed un vertice, al disotto oppure a sinistra del punto in questione, su uno degli assi. Il triangolo così costruito[2] avrà un vertice non appartenente agli assi, e precisamente quello di coordinate (*n* - *m*, *m*), se *n* > *m*; e (*m* - *n*, *n*) se é *n* < *m*.
Se il nuovo punto sta sulla diagonale del primo quadrante, il valore comune delle sue coordinate fornirà il MCD cercato e l'algoritmo sarà terminato; altrimenti si ripete la costruzione precedente, che diventa, a sua volta, un algoritmo geometrico, fino ad ottenere un punto sulla diagonale, cioè un punto avente l'ascissa uguale all'ordinata. le cui coordinate forniscono il MCD cercato. Considerando, per esempio, la coppia (14, 8) si ha il diagramma:



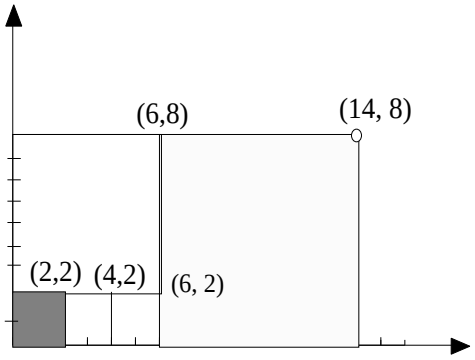
x	*y*
14	8
6	8
2	6
2	4
2	2

MCD (14, 8) = 2

II variazione

[2] Cfr. G. Cesare Barozzi, L'algoritmo euclideo per il calcolo del massimo comune divisore: quattro variazioni facili su tema classico, *Archimede*, 1986, pp. 79-90.

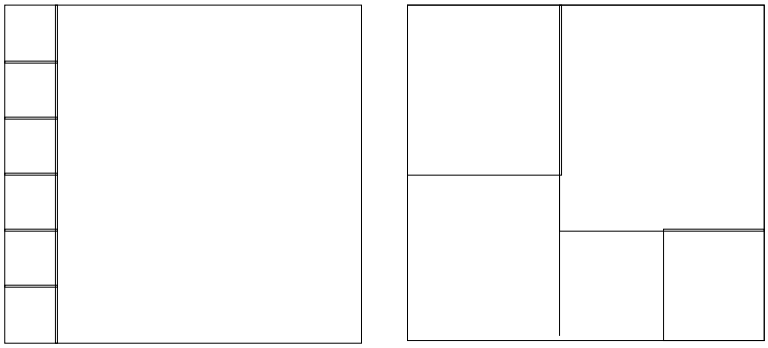
Questa interpretazione geometrica diventa altamente suggestiva considerando che il triangolo rettangolo isoscele che è stato usato è la metà di un quadrato. Si può allora considerare un rettangolo che ha un vertice nell'origine del piano cartesiano e quello opposto nel punto (*n*, *m*).



x	*y*
14	8
6	8
6	2
4	2
2	2

MCD (14, 8) = 2

Se il rettangolo è un quadrato, cioè, se $n = m$, allora non c'è alcun problema da risolvere.
Se, invece, $n \pi m$, si toglie dal rettangolo il più grande quadrato possibile, avente un vertice nel punto (*n, m*): così facendo, il rettangolo di partenza viene diviso in un quadrato e in un rettangolo avente come vertice opposto all'origine il punto di coordinate ($n - m$, *m*), se $n > m$; mentre, se $n < m$ il vertice opposto all'origine avrà come coordinate (*n*, $m - n$).
Nuovamente, se il rettangolo ottenuto è un quadrato, allora il suo lato fornirà il valore del MCD dei due numeri. Se non è un quadrato, si procederà nello stesso modo visto prima, fino ad ottenere un quadrato.
In definitiva, si perviene ad una *pavimentazione* del rettangolo iniziale mediante quadrati, l'ultimo dei quali ha il lato uguale al MCD delle dimensioni del rettangolo iniziale.
Inoltre, quest'ultimo quadrato è il più grande quadrato, a lati di lunghezza intera, che consenta di pavimentare il rettangolo iniziale in modo regolare.
Ciò fa sorgere la domanda: *l'algoritmo euclideo permette di pavimentare un rettangolo con il minor numero di quadrati*?
Ebbene, la risposta è negativa. Infatti, per esempio, l'algoritmo euclideo applicato ad un rettangolo di lati *7* e *6* permette di pavimentarlo con *7* quadrati, ma il rettangolo dato può anche essere pavimentato con *5* quadrati, come viene mostrato nella figura seguente:



III variazione

Come s'è visto, l'algoritmo euclideo delle divisioni successive ci fornisce pure un metodo per determinare il resto della divisione di un numero per un altro. Infatti, dati due numeri, *n* ed *m*, con $n > m$, se occorrono *q* sottrazioni del numero *m* per scendere al di sotto di *m*, ciò significa che *q* è l'intero positivo per cui

$$0 \leq n - qm < m$$

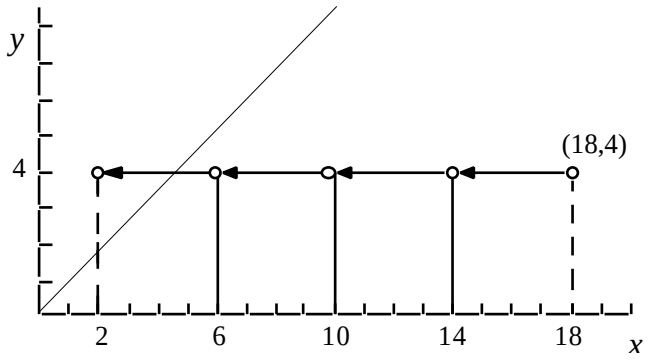
cioè, *q* è il quoziente della divisione di *n* per *m* ed $r = n - mq$ è il resto della divisione stessa.
Per esempio, con $n = 13$ ed $m = 5$, applicando l'algoritmo euclideo otteniamo:

n	*m*
13	5
8	5
3	5
3	2
1	2
1	1

Se osserviamo la tabella dei valori ottenuti, ci accorgiamo che in realtà non abbiamo fatto altro che sottrarre *5* per due volte da *13* fino ad ottenere il valore *3*, che è minore di *5*; dunque:

$$13 = 2 \cdot 5 + 3$$

Abbiamo, quindi, eseguito la divisione di *13* per *5*, ottenendo il quoziente *2* e il resto *3*.
Della divisione euclidea si può fornire un'interpretazione geometrica nel modo seguente. A partire dal punto di coordinate (n, m), ci si sposta parallelamente all'asse delle ascisse nel senso delle *x* decrescenti, descrivendo tratti di lunghezza *m*, fino a raggiungere la regione al di sopra della «prima bisettrice degli assi».
L'ascissa del punto finale è il resto cercato della divisione di *n* per *m*, mentre il numero dei segmenti orizzontali è il quoziente. Considerando, per esempio, la coppia (18, 4) si ha il diagramma:



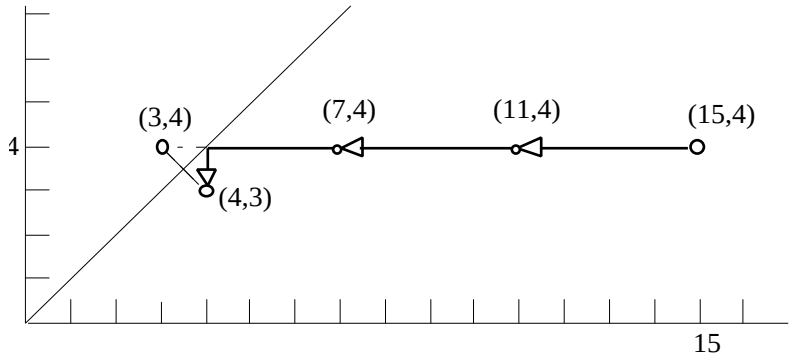
IV variazione

Abbiamo constatato che l'algoritmo euclideo delle divisioni successive riduce il calcolo del MCD tra due numeri *n* ed *m* alla generazione ricorsiva di un numero finito di coppie mediante la regola di trasformazione:

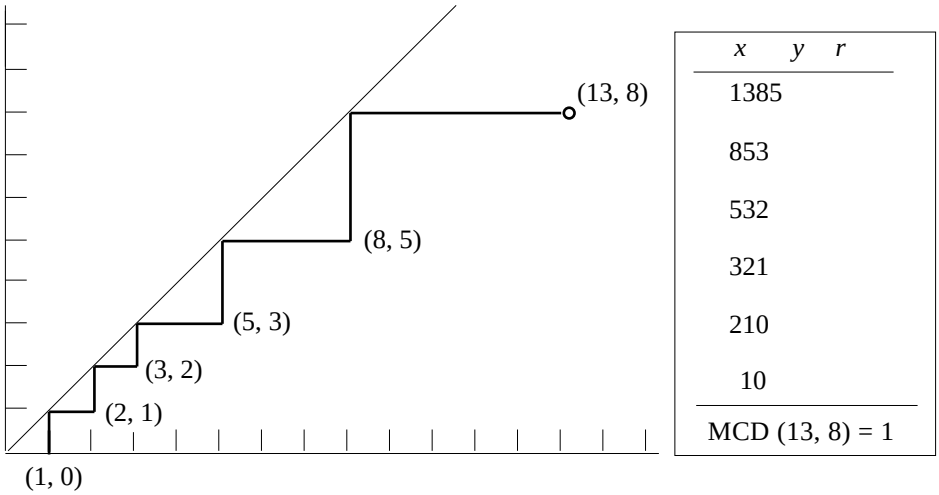
$$(n, m) \rightarrow (m, r)$$

dove *r* è il resto della divisione di *n* per *m*, $(n > m)$, fino a pervenire ad una coppia finale con la seconda coordinata nulla, per cui l'altra coordinata non nulla rappresenterà il MCD cercato.

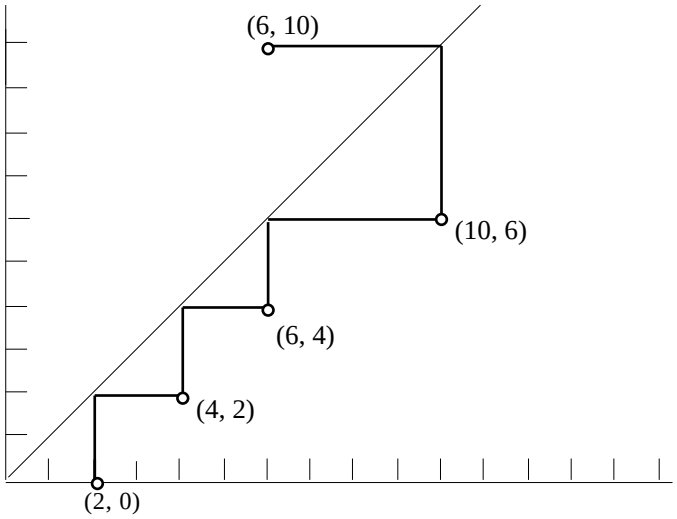
Abbiamo visto come visualizzare la ricerca del resto della divisione di *n* per *m*, cioè come passare dalla coppia (*n, m*) alla coppia (*r, m*); ebbene, il punto (*m, r*) che interessa a noi è allora il simmetrico del punto (*r, m*) rispetto alla diagonale del primo quadrante degli assi.
Possiamo allora pensare di ottenerlo, dal grafico precedente, immaginando che il punto mobile, partito da (*n, m*), una volta raggiunta la diagonale «rimbalzi» su quest'ultima, dirigendosi verso il basso fino a raggiungere il punto (*m, r*).
In definitiva, otteniamo una spezzata di due lati, che congiungono il punto (*n, m*) al punto (*m, r*), e aventi il vertice in comune nel punto (*m, m*). Così, per esempio, con la coppia (15, 4) si ha il diagramma:



Si può, allora, visualizzare l'intero algoritmo euclideo per la ricerca del MCD tra due numeri naturali *n* ed *m* mediante la spezzata data dall'unione delle spezzate di due lati che congiungono ciascun punto con il successivo, fino a raggiungere un punto che giace sull'asse delle ascisse e la cui ascissa fornisce il MCD cercato. Così, per esempio, nel caso della coppia (*13, 8*) avremo la seguente spezzata:



Nel caso in cui sia $n < m$, allora si scambierà il punto (*n, m*) con il suo simmetrico rispetto alla diagonale, dopodiché tutto riprenderà come prima.



x	*y*	*r*
6	10	6
10	6	4
6	4	2
4	2	0
2	0	

MCD (6, 10) = 2

f) Il massimo comun divisore e i numeri di Fibonacci

Possiamo utilizzare quest'ultima rappresentazione del MCD di due numeri per mostrare che *la lunghezza dell'algoritmo euclideo applicato alla coppia formata da due numeri successivi di Fibonacci,* (f_n, f_{n+1})*, è n - 1.*

I numeri di Fibonacci possono essere introdotti in un contesto didattico in svariati modi, ma, forse, la versione più interessante è ancora quella originale, presentata dal grande matematico medievale Leonardo Pisano, detto il Fibonacci, nel suo *Liber Abaci* del 1202.

Fibonacci

Tutto ha inizio da un problema di conigli:

Quante coppie di conigli si otterranno in un anno da una coppia, supponendo che ogni coppia produca ogni mese una nuova coppia la quale sia in grado di produrre un'altra coppia dal secondo mese?

Esclusi i casi di morte, la soluzione cercata si ottiene osservando che il primo mese si ha evidentemente una sola coppia di conigli, quella di partenza; il secondo mese si avrà ancora la stessa coppia, che ora può generarne un'altra, essendo diventata prolifica; il terzo mese si

avranno due coppie; il quarto mese, alle due coppie se ne aggiungerà una terza, generata dalla coppia di partenza; l'aumento delle coppie avviene secondo lo schema della tabella seguente:

Gennaio	1	Luglio	13
Febbraio	1	Agosto	21
Marzo	2	Settembre	34
Aprile	3	Ottobre	55
Maggio	5	Novembre	89
Giugno	8	Dicembre	144

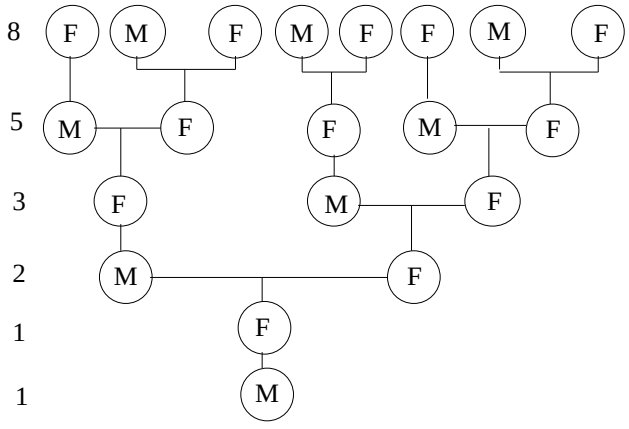
Si potrebbe pensare che il problema dei conigli sia un po' artificiale, e che in questo modo la successione di Fibonacci, pur essendo interessante, rappresenta solo una pura curiosità matematica, a livello di un gioco di società.
La successione di Fibonacci, com'è universalmente nota, è quindi la seguente:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, . . .

Essa può essere quindi definita dalla *formula ricorsiva*:

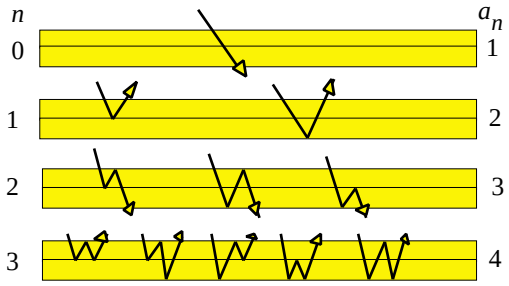
$$F_0=1;$$
$$F_1=1;$$
$$F_n=F_{n-1}+F_{n-2} \qquad (n>1)$$

Ma i numeri di Fibonacci possono sorgere anche in maniera naturale. Consideriamo, per esempio, l'albero genealogico di un'ape maschio, che si chiama fuco. Ogni fuco viene generato asessualmente da un'ape femmina (l'ape regina); ogni femmina ha però due genitori, un maschio ed una femmina, per cui si ha la seguente genealogia:

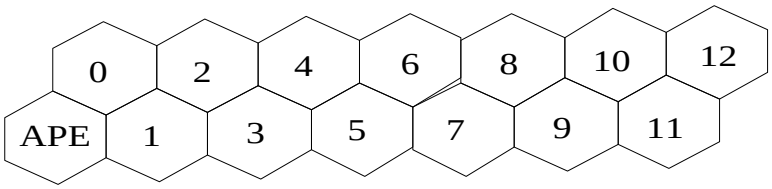


Un fuco ha quindi un nonno ed una nonna, un bisnonno e due bisnonne, due trisnonni e tre trisnonne, e così via; per cui, in generale, per induzione, alla n-esima generazione avrà F_{n+1} antenati, se con F_n indichiamo l'n-esimo termine della successione.
Un altro esempio del modo in cui si incontra la successione di Fibonacci si ha dallo studio, realizzato dal fisico-matematico americano Leo Moser nel 1963: supponiamo di mettere due lastre di vetro una sull'altra; quanti modi a_n esistono affinché un raggio di luce attraversi le due

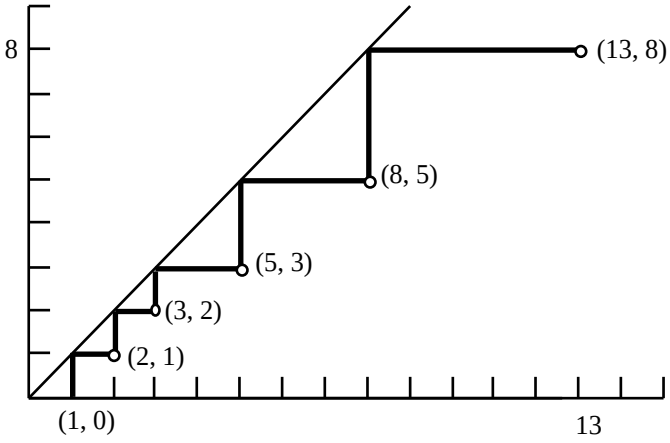
lastre di vetro o venga riflesso, cambiando la direzione n volte? I primi casi si hanno per $n=0,1,2,3$ come è mostrato nella figura.



Se un raggio non viene riflesso, cioè quando $n=0$, allora esso attraverserà le lastre in un unico modo; per n pari il raggio rimbalza un numero pari di volte e passa attraverso i vetri; quando invece n è dispari, il raggio è riflesso e riemerge dalla stessa parte da cui è giunto. Gli a_n sembrano essere proprio i numeri di Fibonacci. Osservando la figura possiamo rendercene conto; infatti, per $n \geq 2$ i raggi con n rimbalzi possono comportarsi in due modi: o effettuano il loro primo rimbalzo sulla superficie opposta e continuano in a_{n-1} modi, oppure iniziano rimbalzando sulla superficie mediana e poi rimbalzano ancora indietro, terminando in a_{n-2} modi. Si ha quindi la regola che governa la successione dei numeri di Fibonacci: $a_n=a_{n-1}+a_{n-2}$, con le condizioni iniziali leggermente differenti, perché in questo caso $a_0=1=F_1$ e $a_1=2=F_2$; ciò vuol dire che ogni cosa è semplicemente spostata di un posto rispetto alla successione di Fibonacci, per cui $a_n=F_{n+1}$.
In modo molto simile si può applicare la successione di Fibonacci per determinare il numero di percorsi che un'ape può scegliere per spostarsi attraverso un'arnia di celle esagonali:



Le celle si estendono all'infinito verso destra e si suppone che l'ape si possa muovere sempre solo verso una cella adiacente e sempre verso destra. È allora chiaro che per raggiungere la cella 0 c'è un solo percorso, per raggiungere la cella 1 ve ne sono due, tre per raggiungere la cella 2, cinque per la cella 3, e così via. Quindi, il numero dei percorsi è F_{n+1}, se n è il numero delle celle considerate.
Quattro secoli più tardi, nell'opera *De nive sexangula* del 1611, Kepler mise in evidenza la proprietà più nota di tale successione, che sarà stata osservata certamente dallo stesso Leonardo Pisano, cioè che ogni termine è dato dalla somma dei due termini precedenti. Tornando al MCD tra due numeri di Fibonacci si può verificare, come s'è detto all'inizio, che, per ogni n naturale, il MCD tra due numeri successivi di Fibonacci è 1, e che la lunghezza dell'algoritmo euclideo è di $n - 1$ passi. Per esempio, considerando i numeri $f_7 = 13$ e $f_6 = 8$ si ha:



f_7	f_6
13	8
8	5
5	3
3	2
2	1
1	0

MCD (13, 8) = 1

g) L'algoritmo euclideo come gioco

L'idea, interessante per i suoi risvolti didattici, di presentare l'algoritmo euclideo come un gioco da giocare in coppia si deve a due matematici americani, A.J. Cole e A.J.T. Davie[3], in un lavoro pubblicato nel 1969.

Alla base della loro idea c'è la nozione di *insieme derivato di un dato insieme* e di *successione derivata*.

Il punto di partenza è il seguente: consideriamo la successione di insiemi (formati anche da due elementi uguali):

$\{78, 35\} \rightarrow \{43, 35\} \rightarrow \{8, 35\} \rightarrow \{8, 11\} \rightarrow \{8, 3\} \rightarrow \{2, 3\} \rightarrow \{2, 1\} \rightarrow$

$\rightarrow \{0, 1\}$.

Ciascun insieme è ottenuto dal precedente sottraendo qualche multiplo intero positivo di uno dei suoi elementi dall'altro. Così, nel passaggio $\{78, 35\} \rightarrow \{43, 35\}$, il secondo insieme è stato ottenuto dal primo sottraendo *35* da *78*; mentre nel passaggio $\{8, 35\} \rightarrow \{8, 11\}$, il secondo insieme è stato ottenuto dal primo sottraendo $24 = 3 \cdot 8$ da *35*.

Ebbene, quando un insieme $\{a, b\}$, di interi non negativi, viene fuori, in qualche modo, da un altro insieme $\{m, n\}$, allora si dice che esso è un *insieme derivato* di $\{m, n\}$. Una successione, come quella scritta sopra, di insiemi derivati l'uno dall'altro, e in cui l'ultimo insieme contiene lo zero, verrà detta *successione derivata.*

Se $\{a, b\}$ è un insieme derivato di $\{m, n\}$, con il minimo valore per la somma $a + b$, allora esso verrà detto un *insieme derivato minimo* di $\{m, n\}$.

Per esempio, nella successione di prima, l'insieme $\{43, 35\}$ non è un insieme derivato minimo di $\{78, 35\}$, mentre $\{2, 3\}$ è un insieme derivato minimo di $\{8, 3\}$.

Il passaggio da un insieme ad un insieme derivato è una *mossa* del gioco, e il passaggio ad un insieme che un elemento uguale a zero è una *mossa vincente*.

1) Notando che $\{m, n\} = \{n, m\}$ per tutti gli m ed n, si ha che:

i) $\{m, n\}$ ha t insiemi derivati, dove t è il più grande intero positivo per cui risulta $tm \leq n$;

[3] A.J. Cole, A.J.T. Davie, *A game based on the Euclidean algorithm and a winning strategy for it*, Mathematical Gazette, LIII (1969), pp. 354-357, I, 1, 2, 3.

Per esempio, l'insieme $\{89, 14\}$ ha 6 insiemi derivati, perché 6 è il più grande intero positivo per cui risulta $6 \cdot 14 < 89$.

ii) $\{m, n\}$ ha esattamente un minimo insieme derivato, che è $\{m, n - tm\}$, con t tale che $tm \leq n$; Per esempio, $\{89, 14\}$ ha l'unico minimo insieme derivato $\{14, 89 - 6 \cdot 14\} = \{14, 5\}$.

iii) se $\{a, b\}$ è un insieme derivato di $\{m, n\}$, allora il massimo comun divisore di a e b è uguale al massimo comun divisore di m ed n; in simboli $(a, b) = (m, n)$.

iv) ogni successione derivata che inizia con $\{m, n\}$, finisce con $\{0, (m, n)\}$.

2) Se due giocatori, *A* e *B*, iniziano con un insieme qualsiasi $\{m, n\}$ e alternativamente muovono, dando origine ad una successione derivata, *A* per primo e *B* per secondo, desiderando ciascuno fare la mossa vincente, allora il gioco che ne vieni fuori si chiama "*il gioco di Euclide*". Ebbene:

i) se in una qualsiasi fase del gioco, si ottiene un insieme in cui un elemento è un multiplo intero positivo dell'altro, allora il giocatore che deve muovere può vincere ottenendo l'insieme derivato minimo. Per esempio, se viene lasciato l'insieme $\{10, 5\}$, il giocatore che deve fare la mossa seguente vincerà ottenendo l'insieme minimo $\{5, 0\}$.

ii) non è sempre vantaggioso per un giocatore fare una mossa per ottenere un insieme derivato minimo (perché l'altro giocatore può giungere per primo alla mossa vincente).

iii) se vi è una strategia vincente per *A*, allora ad ogni gioco deve scegliere fra due opportunità: o l'insieme derivato minimo o l'insieme derivato il cui unico insieme derivato è l'insieme derivato minimo.

iv) quando $1 < \frac{a}{m} < \tau$ (con $\tau = \frac{1 + \sqrt{5}}{2}$, cioè il *rapporto aureo*) vi è un'unica mossa da fare partendo dall'insieme $\{a, m\}$, cioè passare all'insieme $\{r, m\}$ dove

$$\frac{m}{r} > \tau.$$

3) *i*) Il giocatore che muove per primo, partendo da $\{m, n\}$, con $0 < m < n$, può forzare la vittoria per se stesso se e solo se $\frac{m}{r} > \tau$.

ii) quando inizia un gioco con $\{m, n\}$, allora il giocatore *A* può forzare la vittoria se $\frac{n}{m} = 1$ oppure $\frac{n}{m} > \tau$, mentre se non c'è alcuna di queste possibilità, allora la vittoria può essere forzata dal giocatore *B*.

4) Basandosi sulle nozioni di insieme derivato minimo e di successione derivata, l'algoritmo euclideo per trovare il massimo comun divisore tra due numeri interi positivi a e b può essere descritto come una successione derivata che inizia con l'insieme $\{a, b\}$ e in cui ciascun altro elemento della successione è il minimo insieme derivato di quello precedente. Così, se $a > b$, ed $a = q\,b + r$, $0 \leq r < b$ (q ed r interi), la prima mossa sarebbe $\{a, b\} \rightarrow \{b, r\}$.

h) Alcune applicazioni dell'Algoritmo euclideo

L'algoritmo euclideo è usato in Matematica non solo per lo scopo originario per cui esso è stato strutturato, ma anche in moltissimi altri casi, come, per esempio, partendo dall'antichità, nel metodo usato dal matematico persiano al-Khayyam per confrontare tra loro due rapporti o per dimostrare la loro uguaglianza; nella risoluzione delle equazioni indeterminate; nella teoria delle frazioni continue studiate sistematicamente studiate da Leonhard Euler; e anche, ciò che può sembrare sorprendente, nel metodo di Sturm per determinare il numero delle radici reali di un'equazione algebrica. Noi accenneremo a quelle applicazioni che possono dare qualche spunto didattico.

1] Determinazione del minimo comune multiplo mediante il massimo comun divisore

Una prima, semplice applicazione dell'algoritmo euclideo per il massimo comun divisore di due numeri, è la determinazione del *minimo comune multiplo* degli stessi numeri.
Il minimo comune multiplo di due numeri è il più piccolo multiplo che sia divisibile per entrambi i numeri. Così, per esempio, il minimo comune multiplo tra *12* e *10* è *60*, perché, se scriviamo i multipli di *12* e quelli di *10* il più piccolo multiplo comune ai due numeri in cui ci imbattiamo è proprio *60*:

12	12	24	36	48	**60**	⋯	
10	10	20	30	40	50	**60**	⋯

Il minimo comune multiplo di due numeri a e b, si suole indicare con $[a, b]$.
Del minimo comune multiplo di due numeri si dimostra la proprietà seguente:

Ogni multiplo comune di a e b è divisibile per il loro minimo comune multiplo.

Ebbene, il minimo comune multiplo di due numeri può anche essere calcolato mediante il massimo comun divisore dei due numeri. Si dimostra, cioè, il

Teorema

Se $d = (a, b)$ è il massimo comun divisore di a e b, allora il minimo comune multiplo di a e b è dato da:

$$[a, b] = \frac{a\,b}{d} \ .$$

Dimostrazione

La proprietà è vera se i numeri a e b sono primi fra loro; infatti, in questo caso si ha che

$$(a, b) = 1 \text{ e } [a, b] = a \cdot b$$

per cui

$$(a, b) \cdot [a, b] = a \cdot b.$$

Se $(a, b) \neq 1$, allora potremo scrivere i due numeri come multipli del loro massimo comun divisore $d = (a, b)$:

$$a = a_1 \cdot d$$
$$b = b_1 \cdot d$$

Allora si avrà:

$$[a, b] = [\, a_1 \cdot d, b_1 \cdot d] = d \cdot [a_1, b_1] = d \cdot (a_1 \cdot b_1)$$

Cioè

$$[a, b] = d \cdot (a_1 \cdot b_1)$$

Errore. $[a, b] = d \cdot (a_1 \cdot b_1) \cdot = EQ \F((d \cdot a1) \cdot (d \cdot b1);d) = EQ \F(a \cdot b;d)$

Per determinare il massimo comun divisore e il minimo comune multiplo di più di due numeri, basta osservare che:

1] Se d = (a, b, c) allora esso dovrà dividere sia (a, b) che c, per cui si avrà:

$$d = ((a, b), c)$$

2] Se m = [a, b, c], dovendo essere divisibile sia per [a, b] che per c dovrà risultare:

$$m = [[a, b], c].$$

L'algoritmo seguente riesce a calcolare il massimo comun divisore e il minimo comune multiplo di due numeri a e b «in parallelo»:

0. *X* *a, Y* *b*

1. U ⟵ X, V ⟵ Y
2 finché X ≠ Y, ripetere:

2.1 se X < Y, allora: Y ⟵ Y - X, V ⟵ V + U

2.2 altrimenti: X ⟵ X - Y, U ⟵ U + V

3. stampare $\frac{X + Y}{2}$, $\frac{U + V}{2}$

4. fine

2] Massimo comun divisore di due numeri come combinazione lineare dei numeri dati

Se $d = (a, b)$, allora si possono trovare due numeri interi, positivi o negativi, k ed l, tali che si abbia: $d = k\,a + l\,b$.

Dimostrazione

Scriviamo la successione delle divisioni che ci permettono di trovare il massimo comun divisore noto:

$$\begin{aligned} a &= bq_1 + r_1 && (0 < r_1 < b) \\ b &= r_1q_2 + r_2 && (0 < r_2 < r_1) \\ r_1 &= r_2q_3 + r_3 && (0 < r_3 < r_2) \\ r_2 &= r_3q_4 + r_4 && (0 < r_4 < r_3) \end{aligned}$$

....

finché si ottiene l'ultimo resto non nullo che coincide con il massimo comun divisore di *a* e *b*. Ebbene, dalla prima di queste uguaglianze si ha:

$$r_1 = a - q_1 b$$

e ciò significa che r_1 può essere scritto nella forma $k_1\ a\ +\ l_1\ b$, perché basta prendere, in questo caso, $k_1 = 1$ e $l_1 = - q_1$.
In virtù di questa uguaglianza, dalla seconda uguaglianza si ottiene:

$$r_2 = b - q_2 r_1 = b - q_2\ (k_1\ a + l_1\ b) = (\ - q_2 k_1)\ a + (1 - q_2 l_1)\ b = k_2\ a + l_2\ b.$$

Evidentemente, anche i successivi resti potranno essere espressi in questo modo, per cui si arriverà alla fine ad esprimere il resto *n*-esimo non nullo, r_n, nella forma:

$$\boxed{r_n = k\ a + l\ b}$$

che è quello che si voleva dimostrare. ■

Esempio
Consideriamo il caso del massimo comun divisore tra *84* e *25*, che è *1*. In questo caso le divisioni successive sono:

$$84 = 25 \cdot 3 + 9$$
$$25 = 9 \cdot 2 + 7$$
$$9 = 7 \cdot 1 + 2$$
$$7 = 2 \cdot 3 + 1$$
$$2 = 1 \cdot 2 + 0$$

Dalla prima di queste uguaglianze si ha:

$$9 = 84 - 25 \cdot 3$$

dalla seconda:

$$7 = 25 - 2 \cdot 9 = 25 - 2 \cdot (84 - 25 \cdot 3) = - 2 \cdot 84 + 7 \cdot 25$$

dalla terza:

$$2 = 9 - 1 \cdot 7 = 9 - 1 \cdot (- 2 \cdot 84 + 7 \cdot 25) = 9 + 2 \cdot 84 - 7 \cdot 25 = 84 - 3 \cdot 25 + 2 \cdot 84 - 7 \cdot 25 = 3 \cdot 84 - 10 \cdot 25$$

e infine dalla quarta:

$$1 = 7 - 2 \cdot 3 = (- 2 \cdot 84 + 7 \cdot 25(- 3 \cdot (3 \cdot 84 - 10 \cdot 25) = - 2 \cdot 84 + 7 \cdot 25 - 9 \cdot 84 + 30 \cdot 25 = - 11 \cdot 84 + 37 \cdot 25.$$

3] Dimostrazione del teorema fondamentale dell'aritmetica.

Il teorema fondamentale dell'Aritmetica può essere enunciato nel modo seguente:

Ogni numero intero positivo maggiore dell'unità ammette una ed una sola scomposizione in fattori primi.

Usando i simboli si direbbe:

Per ogni intero $n > 1$, esiste un insieme finito di numeri primi $p_1 \le p_2 \le p_3 \le \ldots \le p_r$ tali che

$$n = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_r$$

e questa fattorizzazione è unica.

L'importanza di questo teorema nella teoria dei numeri è appunto *fondamentale*, perché essenzialmente preserva tutti i nostri ragionamenti dall'ambiguità. Per dimostrarlo possiamo utilizzare il lemma seguente:

Lemma

Se un numero primo divide un prodotto $a \cdot b$ di due numeri, dev'essere divisore o di a o di b.

Dimostrazione

Supponiamo che il numero primo p non sia divisore di a, allora il massimo comun divisore tra a e p è 1, $(a, p) = 1$. Si potranno trovare, quindi, due interi, k ed l, tali che

$$1 = k \cdot a + l \cdot p$$

Moltiplicando i due membri di questa uguaglianza per b si ottiene:

$$b = kab + lpb.$$

Poiché *per ipotesi* il numero primo p divide il prodotto $a \cdot b$, allora tale prodotto sarà un multiplo di p:

$$a \cdot b = p \cdot r$$

per cui, sostituendo nell'ultima uguaglianza si ottiene:

$$b = kpr + lpb = p \cdot (kr + lb)$$

e quindi *p è divisore di b.* ■

Dimostrazione del Teorema fondamentale

Supponiamo che il numero n ammetta due scomposizioni in fattori primi:

$$n = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_r = q_1 \cdot q_2 \cdot q_3 \cdot \ldots \cdot q_s$$

Poiché p_1 divide n, ed è $n = q_1 \cdot q_2 \cdot q_3 \cdot \ldots \cdot q_s$, allora p_1 divide il prodotto $q_1 \cdot q_2 \cdot q_3 \cdot \ldots \cdot q_s$. Ciò vuol dire, per il lemma precedente, che p_1 dovrà dividere almeno uno dei fattori del prodotto $q_1 \cdot q_2 \cdot q_3 \cdot \ldots \cdot q_s$, per esempio q_k. Ma q_k è un numero primo, per cui dovrà risultare $p_1 = q_k$. Questi due fattori si potranno quindi cancellare. Ripetiamo lo stesso ragionamento per p_2, per cui esso potrà essere cancellato assieme al fattore q_t al quale risulterà uguale. Continuando lo stesso tipo di ragionamento, le p e le q verranno associate in coppie uguali, e ciò vuol dire che, salvo l'ordine dei fattori, le due scomposizioni di n sono uguali. ■

4]Le frazioni continue e la risoluzione delle equazioni diofantine.

L'algoritmo euclideo per la ricerca del massimo comun divisore di due numeri interi ci permette di rappresentare il quoziente di due numeri interi mediante una frazione particolare detta *frazione continua*.

Le frazioni continue furono studiate dai grandi matematici dei secoli decimosettimo e decimottavo, e ancora oggi sono oggetto di studio. Esse - come venne rimarcato da C.D. Olds (*Frazioni continue*, Zanichelli, 1970) - costituiscono uno dei più importanti strumenti per nuove scoperte nella Teoria dei numeri e nel campo delle approssimazioni diofantee. Si può mostrare agli studenti che al concetto di frazione continua ci si può arrivare in modo del tutto naturale, anche con esempi piuttosto semplici.
Applichiamo, per esempio, l'algoritmo euclideo ai numeri *67* e *29*. Otteniamo:

$$67 = 29 \cdot 2 + 9$$
$$29 = 9 \cdot 3 + 2$$
$$9 = 2 \cdot 4 + 1$$
$$2 = 1 \cdot 2 + 0$$

da cui risulta che $(67, 29) = 1$

Combinando queste uguaglianze, possiamo scrivere il numero razionale $\frac{67}{29}$ nella forma:

$$\frac{67}{29} = 2 + \frac{9}{29} =$$
$$= 2 + \frac{1}{\frac{29}{9}} =$$
$$= 2 + \frac{1}{3 + \frac{2}{9}} =$$
$$= 2 + \frac{1}{3 + \frac{1}{\frac{9}{2}}} =$$
$$= 2 + \frac{1}{3 + \frac{1}{4 + \frac{1}{2}}} .$$

La frazione che compare a destra del segno d'uguaglianza si chiama *frazione continua limitata* perché il numero dei quozienti parziali *2, 3, 4, 2* è finito.
Per comodità di scrittura si può rappresentare così:

$$\frac{67}{29} = 2 + \frac{1}{3+}\ \frac{1}{4+}\ \frac{1}{2}$$

oppure nella forma ancora più compatta:

$$\frac{67}{29} = [2, 3, 4, 2]$$

Esistono pure *frazioni continue illimitate*, cioè con un numero infinito di quozienti parziali.

Se consideriamo, ad esempio, il numero irrazionale $\sqrt{2}$ e vogliamo rappresentarlo sotto la forma di una frazione continua, possiamo procedere nel modo seguente. Poiché il più grande intero minore di $\sqrt{2}$ è il numero *1*, allora è lecito scrivere:

$$\sqrt{2} = 1 + \frac{1}{n}$$

da cui ricaviamo $n=\sqrt{2}+1$. Poiché il più grande intero minore di n é *2*, allora si avrà

$$n = 2 + \frac{1}{p}$$

Da essa ricaviamo che

$$p = \frac{1}{n-2} = \frac{1}{\sqrt{2}-1} = \sqrt{2} + 1$$

per cui $n = p$, e continuando nello stesso modo con cui abbiamo calcolato i valori di n e di p troveremo sempre lo stesso valore $\sqrt{2}+1$.
Allora potremo rappresentare $\sqrt{2}$ mediante la frazione continua illimitata:

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ldots}}}$$

oppure come

$$\sqrt{2} = 1 + \frac{1}{2 +}\ \frac{1}{2 +}\ \frac{1}{2 + \ldots} = [1, 2, 2, 2, \ldots] = [1, \bar{2}]$$

Quando si ha a che fare con frazioni continue limitate o illimitate si fa uso della nozione di *frazioni ridotte* o semplicemente *ridotte.*

Riprendiamo lo sviluppo di $\frac{67}{29}$:

$$\frac{67}{29} = 2 + \cfrac{1}{3 + \cfrac{1}{4 + \cfrac{1}{2}}}$$

Le ridotte sono le frazioni che si ottengono arrestandosi al primo, al secondo, al terzo, al quarto quoziente parziale. Nel nostro caso esse sono:

$$\frac{2}{1},\quad 2 + \frac{1}{3} = \frac{7}{3},\quad 2 + \cfrac{1}{3 + \cfrac{1}{4}} = \frac{30}{13},\quad 2 + \cfrac{1}{3 + \cfrac{1}{4 + \cfrac{1}{2}}} = \frac{67}{29}$$

e prendono rispettivamente il nome di *prima*, *seconda*, *terza* e *quarta* ridotta. Evidentemente, l'ultima ridotta coincide con la frazione di partenza. Le ridotte di una frazione continua godono di una proprietà importante. Se

$$\frac{p}{q} = [a_1, a_2, a_3, \ldots, a_{n-1}, a_n]$$

è una generica frazione continua, e le sue ridotte sono:

$$\frac{p_1}{q_1} = \frac{a_1}{1}, \quad \frac{p_2}{q_2} = a_1 + \frac{1}{a_2}, \quad \frac{p_3}{q_3} = a_1 + \frac{1}{a_2 + \frac{1}{a_3}}, \quad \ldots$$

si può dimostrare per induzione che tra i numeratori e i denominatori di esse sussiste la relazione

$$\boxed{p_n\, q_{n-1} - p_{n-1}\, q_n = (-1)^n \quad \text{con } n \geq 0.}$$

Da quanto si è detto risulta evidente che per ottenere una frazione continua si mette in atto un vero e proprio *algoritmo* che utilizza sempre l'algoritmo euclideo per determinare i quozienti successivi che serviranno per scrivere la frazione continua. Se riprendiamo in considerazione la frazione $\frac{67}{29}$ e le successive divisioni

$$67 = 29 \cdot \mathbf{2} + 9$$
$$29 = 9 \cdot \mathbf{3} + 2$$
$$9 = 2 \cdot \mathbf{4} + 1$$
$$2 = 1 \cdot \mathbf{2} + 0$$

ci accorgiamo che queste ultime forniscono immediatamente i divisori necessari; per cui l'algoritmo può essere schematizzato nei passi:

Le frazioni continue possono servire per risolvere le cosiddette equazioni indeterminate di primo grado o *diofantee*, dal nome del grande matematico alessandrino Diofanto, vissuto all'incirca nel III secolo d.C..
Di esempi che conducono a questo tipo di equazioni ve ne è una grande quantità.
Un contadino compra delle piante da frutta a *80.000* lire l'una e delle piante ornamentali a *50.000* l'una. Paga in tutto *810.000* lire. Quante piante da frutta e ornamentali ha comprato?

Se *x* è il numero delle piante da frutta e *y* quello delle piante ornamentali, il problema si traduce nell'equazione di primo grado in due incognite:

$$80\,000\, x + 50\,000\, y = 810\,000$$

ovvero

$$8\, x + 5\, y = 81.$$

Risolvere, quindi, il problema, significa determinare le coppie dei numeri interi positivi che soddisfano l'equazione. Essa è chiaramente un'equazione *indeterminata,* il che significa che possiamo sempre trovare qualche valore di *y* in corrispondenza a qualsiasi valore della *x.*
È importante fare rilevare agli allievi che un'equazione come quella scritta si può risolvere in molti modi. *Infatti non c'è nulla di male se si risolvono tali equazioni per tentativi o con intelligenti congetture.*
Per esempio, scritta l'equazione ottenuta nella forma:

$$81 - 8\, x = 5\, y$$

possiamo cercare i valori interi non negativi di $81 - 8x$ sia multiplo di 5. Se facciamo prendere, allora, alla x, successivamente, i valori $0, 1, 2, 3, 4, \ldots, 10$, troviamo che i soli valori non negativi per cui $81 - 8x$ è un multiplo non negativo di 5 sono $x = 2$ e $x = 7$. Infatti, i calcoli danno i valori:

x	1	$\textbf{2}$	3	4	5	6	$\textbf{7}$	8	9	10
$81 - 8x$	74	$\mathbf{65}$	57	49	41	33	$\mathbf{25}$	17	9	1
y	$\frac{74}{5}$	$\mathbf{13}$	$\frac{57}{5}$	$\frac{49}{5}$	$\frac{41}{5}$	$\frac{33}{5}$	$\mathbf{5}$	$\frac{\mathbf{17}}{\mathbf{5}}$	$\frac{\mathbf{9}}{\mathbf{5}}$	$\frac{\mathbf{1}}{\mathbf{5}}$

Il problema ha quindi le due soluzioni intere $(2, 13)$ e $(7, 5)$. Questo metodo può essere, forse, il più naturale per un allievo che incontra per la prima volta un problema di questo genere, per cui ha una sua validità didattica, e può servire per avviarlo verso una riflessione ulteriore per cercare un metodo alternativo e più generale che elimini i calcoli laboriosi. Uno di tali metodi utilizza proprio le frazioni continue.

a) Metodo delle ridotte di una frazione continua

Scriviamo una generica equazione diofantea di primo grado:

$$ax + by = c$$

di cui si cercano soluzioni intere x ed y.
Innanzitutto, *dimostriamo che l'equazione data ha una soluzione se e solo se c è multiplo del massimo comun divisore di a e b*.
Infatti, se $d = (a, b)$, allora sappiamo già trovare due interi k ed l tali che

$$d = ka + lb$$

Allora, nel caso in cui $c = d$ l'equazione data ammette la soluzione particolare $x = k$ e $y = l$. Più in generale, se c è multiplo di d, cioè se $c = d \cdot q$, allora si avrebbe:

$$a\,(kq) + b\,(lq) = dq = c$$

per cui l'equazione data avrebbe la soluzione:

$$x = k\,q$$
$$y = l\,q.$$

Viceversa, se, per un dato c, l'equazione data ammette una soluzione (x, y), allora c dev'essere un multiplo di $d = (a, b)$. Infatti d è divisore sia di a che di b, per cui dev'essere divisore anche di c.

Supponendo, ora, a e b primi tra loro, $(a, b) = 1$, si sviluppi il quoziente $\frac{a}{b}$ in frazione continua, e siano $\frac{p_{n-1}}{q_{n-1}}$ e $\frac{p_n}{q_n}$ le due ultime ridotte. Allora risulterà:

$$\frac{p_n}{q_n} = \frac{a}{b}$$

$$p_n \cdot q_{n-1} - p_{n-1} \cdot q_n = (-1)^n$$

e poiché $p_n = a$, $q_n = b$:

$$a \cdot q_{n-1} - b \cdot p_{n-1} = (-1)^n$$

Premesso ciò, si consideri il sistema:

$$\begin{aligned} ax + by &= c \\ p_{n-1} \cdot x + q_{n-1} \cdot y &= k \end{aligned}$$

dove k è un intero qualunque. Risolvendo si ha:

$$x = \frac{c \cdot q_{n-1} - k \cdot b}{a \cdot q_{n-1} - b \cdot p_{n-1}}$$

$$y = \frac{k \cdot a - c \cdot p_{n-1}}{a \cdot q_{n-1} - b \cdot p_{n-1}}$$

e tenendo conto della relazione $a \cdot q_{n-1} - b \cdot p_{n-1} = (-1)^n$ risulterà:

$$x = c \cdot q_{n-1} - k \cdot b$$

$$y = k \cdot a - c \cdot p_{n-1}$$

se n è pari, o ai loro contrari se n è dispari.
Questo metodo elegante per risolvere l'equazione $ax + by = c$ è dovuto a Lagrange, ed apparve nell'*Algebra* di L. Euler, del *1770*, curata dal grande matematico italo-francese.
Risolviamo per esempio l'equazione

$$17x + 13y = 300$$

In questo caso, si ha

$$\frac{17}{13} = [1, 3, 4]$$

per cui la penultima ridotta è:

$$\frac{p_2}{q_2} = \frac{4}{3}$$

per cui si avrà:

$$a \cdot q_{n-1} - b \cdot p_{n-1} = (-1)^n = 17 \cdot 3 - 13 \cdot 4 = -1$$

Le soluzioni dell'equazione data saranno allora:

$$\begin{aligned} x &= 13k - 900 \\ y &= 1200 - 17k \qquad (k = 0, \pm 1, \pm 2, \pm 3, \ldots) \end{aligned}$$

Si noti che il metodo di Lagrange costituisce a tutti gli effetti un *algoritmo* per la risoluzione di un'equazione indeterminata di primo grado a due incognite.

b) Metodo basato sull'algoritmo euclideo

Un altro metodo alternativo di risoluzione di un'equazione del tipo $ax + by = c$ è quello basato sul massimo comun divisore di a e b.

Con questo metodo si determina innanzitutto una soluzione particolare dell'equazione data basandosi sul fatto, già ricordato prima, che se il massimo comun divisore di *a* e *b* divide anche *c*, allora una soluzione particolare dell'equazione è:

$$x^* = k\,q,\ y^* = l\,q.$$

Ora, se $x = x'$ e $y = y'$ è una qualsiasi altra soluzione diversa da x^* e y^*, allora $x = x' - x^*$ e $y = y' - y^*$ è una soluzione dell'equazione "omogenea"

$$ax + by = 0.$$

Infatti, se

$$a\,x' + b\,y' = c$$
$$a\,x^* + b\,x^* = c$$

sottraendo la seconda equazione dalla prima, si ha:

$$a\,(x' - x^*) + b\,(y' - y^*) = 0.$$

Poiché la soluzione più generale dell'equazione $ax + by = 0$ è data da

$$x = \frac{k\,b}{(a,b)}$$
$$y = -\frac{k\,a}{(a;b)}$$

dove *k* è un numero intero, segue immediatamente che la soluzione più generale dell'equazione $ax + by = c$ *sarà data da:*

$$x = x^* + \frac{k\,b}{(a,b)}$$
$$y = y^* - \frac{k\,a}{(a;b)}\ .$$

c) Metodo di Euler

L'idea centrale del metodo usato da Euler per risolvere l'equazione $ax + by = c$, è di far vedere che le sue soluzioni intere sono legate alle soluzioni intere di una equazione analoga, ma con coefficienti minori.
Illustriamo il metodo con l'esempio già considerato dell'equazione $8x + 5y = 81$. Risolviamo l'equazione rispetto alla variabile che ha il coefficiente minore, quindi rispetto alla *y*:

$$y = \frac{81 - 8x}{5}$$

Mettiamo in evidenza i più grandi multipli di *5* contenuti in *81* e *8*:

$$81 = 5 \cdot 16 + 1$$
$$8 = 5 \cdot 1 + 3$$

per cui si potrà scrivere:

$$y = \frac{(5 \cdot 16 + 1) - (5 \cdot 1 + 3)x}{5} = (16 - x) + \frac{1 - 3x}{5} = (16 - x) + t$$

dove $t = \frac{1 - 3x}{5}$, ovvero $3x + 5t = 1$.
Poiché x e y devono essere interi, allora t dev'essere intero, per cui bisogna trovare due interi x e t che soddisfino l'equazione $3x + 5t = 1$. Applichiamo a questa equazione lo stesso procedimento mediante il quale l'abbiamo ottenuta dall'equazione di partenza. Si ha:

$$x = \frac{1 - 5t}{3} = \frac{1 - (2 \cdot 3 - 1)t}{3} = -2t + \frac{t + 1}{3} = -2t + u$$

in cui

$$u = \frac{t + 1}{3}$$

o anche $t = 3u - 1$.
Poiché x e t devono essere interi, dovrà esserlo; pure u. Viceversa, se u è intero, l'ultima relazione mostra che $t = 3u - 1$ è intero. Ma x è pure intero; infatti, essendo $x = -2t + u$, sostituendo il valore di t si ha:

$$x = -2t + u = -2(3u - 1) + u = 2 - 5u.$$

Sostituendo i valori di x e di t nell'espressione della y, si ottiene:

$$y = (16 - x) + t = 16 - (2 - 5u) + 3u - 1 = 8u + 13.$$

quindi, y è intero. In definitiva, la soluzione generale dell'equazione data è:

$$\begin{aligned} x &= 2 - 5u \\ y &= 8u + 13. \end{aligned} \qquad (u = 0, \pm 1, \pm 2, \pm 3, \ldots)$$

La tabella seguente contiene alcune delle infinite soluzioni dell'equazione data.

u	-2	-1	0	1	2	3
x	12	7	2	-3	-8	-13
y	-3	5	13	21	29	37

Numeri primi

> *I matematici hanno tentato invano fino ad oggi di scoprire qualche ordine nella successione dei numeri primi, e abbiamo ragione di credere che essa è un mistero in cui la mente umana non penetrerà mai. Per convincerci di ciò, ci basta dare solo un'occhiata ad alcune tavole di primi che alcuni si sono presi la briga di calcolare oltre un centinaio di migliaia, e ci accorgeremo subito che non vi regna né alcun ordine né alcuna regola.*
>
> Leonhard Euler, *Opere*, ser. 1, v. *2*, p. *241*.

> *Si sa che il problema di distinguere tra numeri primi e numeri composti, e la scomposizione di questi utimi nei loro fattori primi, è uno dei più importanti e utili di tutta l'Aritmetica [...]. La dignità della scienza sembra richiederci la ricerca accurata di tutti gli strumenti necessari per risolvere questo problema così elegante e famoso.*
>
> C.F. Gauss, *Disquisitiones Arithmeticae*, 1801.

I numeri primi costituiscono uno dei campi di ricerca più affascinanti non solo della teoria dei numeri ma dell'intera Matematica. La loro definizione è molto semplice (un numero è primo se è divisibile *soltanto* per se stesso e l'unità), ma i problemi a cui hanno dato luogo sono per la maggior parte profondi e difficili. Essi sono, in un certo senso, gli ‘atomi’ sui quali è costruita tutta la teoria dei numeri.
A dispetto della difficoltà dei problemi che pongono, i numeri primi hanno un fascino al quale difficilmente si può resistere.

a] L'infinità dei numeri primi

I numeri primi inferiori a *100*, ordinati secondo valori crescenti, sono rappresentati dalla tabella seguente:

2 3 5 7 11 13 17 19 23 29 31 37 41

43 47 53 59 61 67 71 73 79 83 89 97 . . .

Come si nota, i numeri primi vanno rarefacendosi al crescere del loro valore. Non è dunque peregrino chiedersi se per caso, da un certo punto in poi, essi scompaiano del tutto, ovvero se non ci sia un numero primo massimo, che possiamo denotare con p_{max}, tale che ogni altro numero naturale $n > p_{max}$ sia composto. In altre parole, sorge spontanea la domanda: *'I numeri primi sono finiti?'* Ebbene, la risposta a questa domanda è negativa, perché i numeri primi sono infiniti, e la prima dimostrazione di ciò si trova nella *Proposizione 20* del *Libro IX* degli *Elementi* di Euclide. Per la sua semplicità e concisione, essa rimane un esempio di eleganza nella storia della Matematica. Supponendo che il numero dei primi sia finito, consideriamo il numero *n* ottenuto sommando l'unità al prodotto dei primi, dei quali il maggiore sia *p*:

$$n = 2 \cdot 3 \cdot 5 \cdot 7 \cdot ... \cdot p + 1$$

Ebbene, *n* non può essere divisibile per nessuno dei primi considerati *2, 3, 5, 7, ... , p* perché se venisse diviso per uno qualsiasi di essi, si otterrebbe come resto l'unità. Ciò vuol dire che vi sono due possibilità:
1] *n* è composto, allora sarà divisibile per qualche primo diverso da quelli considerati, per cui oltre questi primi ve n'è almeno uno in più;
2] *n* non è divisibile per nessun numero minore di se stesso, per cui è primo.
Questa dimostrazione euclidea, benché indiretta, può essere facilmente modificata in modo da fornirci un metodo per costruire, almeno in teoria, una successione di primi. Illustriamolo con un esempio: supponiamo di conoscere i due numeri primi *2* e *3*; a partire da essi vogliamo costruire 5 numeri primi. Consideriamo il numero dato da $2 \cdot 3 + 1 = 7$ che è primo; abbiamo allora tre primi: *2, 3* e *7*. Consideriamo ora il numero $2 \cdot 3 \cdot 7 + 1 = 43$ che è pure primo, per cui i primi sono ora *2, 3, 7, 43*. Nello stesso modo, consideriamo il numero $2 \cdot 3 \cdot 7 \cdot 43 + 1 = 1807$ che non è primo, perché ammette i fattori primi *13* e *139*. Partendo, quindi, da *2* e *3* abbiamo ottenuto la successione di primi: *2, 3, 7, 13, 43, 139*.

b] La distribuzione dei primi

Uno dei problemi più affascinanti e profondi della teoria dei numeri riguarda la distribuzione dei primi. Come sono distribuiti i primi tra i numeri naturali? Seguono un certo ordine, oppure la loro distribuzione è casuale? Sono queste le domande che sorgono quando si esamina una qualunque tavola di numeri primi. Ma ciò che balza subito agli occhi è proprio la mancanza di qualsiasi regolarità. Per esempio, tra i cento numeri che vengono immediatamente prima di *10.000.000* vi sono *9* primi:

9.999.901	*9.999931*	*9.999.971*
9.999.907	*9.999.937*	*9.999.973*
9.999.929	*9.999.943*	*9.999.991*

mentre tra i cento numeri che seguono immediatamente *10.000.000* vi sono solo due primi:

10.000.019

10.000.079

Nessun matematico ha mai scoperto un formula che permetta di calcolare l'*n*-esimo numero primo, né ce n'è una per determinare il primo che viene immediatamente dopo un dato primo. Infatti, nessuno ha mai trovato una regola semplice per trovare un qualsiasi primo maggiore di un primo dato, e sembra che tali problemi sia lontani dall'essere risolti. Non sembra che ci sia

una ragione perché un numero sia primo mentre un altro non lo è. Per sondare la distribuzione dei primi si usa una funzione $\pi(x)$, che serve a denotare il numero dei primi che sono inferiori ad un dato numero x. Per esempio, alcuni dei valori della funzione sono i seguenti:

$\pi(100) = 25$	$\pi(500) = 95$
$\pi(200) = 46$	$\pi(1100) = 111$
$\pi(300) = 62$	$\pi(1200) = 123$
$\pi(400) = 78$	$\pi(1300) = 138$
.....	.....

Poiché $\pi(x)$ tende ad infinito al tendere di x ad infinito, è interessante calcolare il rapporto $\frac{\pi(x)}{x}$ che misura in che proporzione si trovano i numeri primi fino ad x. Così, per esempio, si ha:

x	$\frac{\pi(x)}{x}$
10	$0,4000$
10^2	$0,2500$
10^3	$0,1680$
10^4	$0,1229$
10^5	$0,0959$
10^6	$0,0785$
10^7	$0,0665$
.....	.....

Quindi, i dati suggeriscono che

$$\frac{\pi(x)}{x} \longrightarrow 0 \text{ quando } x \longrightarrow \infty$$

ovvero che *la densità dei numeri primi è zero*.
Ebbene la prima osservazione importante sulla distribuzione dei primi venne fatta da Gauss, quand'era ancora un ragazzo! Egli osservò che il rapporto $\frac{\pi(x)}{x}$ è approssimativamente uguale a $\frac{1}{\log x}$, e che questa approssimazione migliora sempre più al crescere di x. Il grado di approssimazione è dato dal rapporto:

$$\frac{\frac{\pi(x)}{x}}{\frac{1}{\log x}} = \frac{\pi(x) \cdot \log x}{x}$$

i cui valori, per $x = 1000, x = 1\,000\,000, x = 1\,000\,000\,000$ sono:

x	$\frac{\pi(x)}{x}$	$\frac{1}{\log x}$	$\frac{\pi(x) \cdot \log x}{x}$
10^3	$0,168$	$0,145$	$1,159$

10^6	0,078 498	0,0072 382	1,084
10^9	0,050 847 478	0,048 254 942	1,053
.....	.....	.....	

Fu sulla base di questa verifica empirica che il giovanissimo Gauss ipotizzò che il rapporto $\frac{\pi(x)}{x}$ sia *asintoticamente uguale* a $\frac{1}{\log x}$, intendendo dire, con ciò, che al crescere di x, il rapporto $\frac{\pi(x) \cdot \log x}{x}$ si andrà via via avvicinando ad 1, e che la differenza tra 1 e questo rapporto, per valori di x sufficientemente grandi, può divenire piccola a piacere.

Un francobollo commemorativo di Gauss

Tutto ciò si esprime simbolicamente con il segno ~ :

$$\frac{\pi(x)}{x} \sim \frac{1}{\log x}$$

Il segno ~ non può essere sostituito con il segno = di uguaglianza perché, mentre $\pi(x)$ è sempre un numero intero, $\frac{x}{\log x}$ non è intero. Questo è proprio il teorema dei numeri primi, che può anche essere enunciato simbolicamente scrivendo:

$$\lim_{x \to \infty} \frac{\pi(x) \log x}{x} = 1$$

La cosa più sorprendente di questo teorema è il fatto che esso descriva il comportamento medio della distribuzione dei numeri primi mediante la funzione logaritmica, ed è notevole che concetti matematici apparentemente così lontani siano in realtà connessi così intimamente.
Comunque, anche se l'enunciato del teorema è molto semplice, per dimostrarlo passarono quasi cento anni, perché l'analisi si sviluppasse al punto da permettere nello stesso anno, 1896, a J.Hadamard, a Parigi, e a Ch. De La Vallée-Poussin, a Lovanio, di riuscire a darne una dimostrazione completa e rigorosa. A questa dimostrazione sono stati apportati in seguito vari miglioramenti, soprattutto per semplificarla. Solo nel 1949 Atle Selberg e Paul Erdös

riuscirono a darne la prima dimostrazione *elementare*. Una dimostrazione viene detta elementare quando usa delle tecniche che non facciano intervenire concetti estranei alla teoria in cui si inquadra il teorema che si vuole dimostrare. Ma per fare ciò occorre usare delle tecniche più *ristrette*, che tendono a produrre delle dimostrazioni più complicate.

c] *Il crivello di Eratostene*

Eratostene da Cirene (circa 276-195 a.C.) iniziò i suoi studi a Cirene, li continuò ad Alessandria con il poeta Callimaco, li proseguì ad Atene, dove ascoltò gli insegnamenti degli eredi dell'Accademia e del Liceo, tra cui, sembra, lo stesso Stratone, e quelli dei "nuovi filosofi", in particolare di Zenone di Cizio.

Eratostene secondo un antico bassorilievo

Eratostene fu uomo di vasta cultura, sia in campo letterario che in quello matematico-scientifico. Non a caso venne scelto da Tolomeo III per educare il figlio, il futuro Tolomeo IV Filopatore. Si narra che al suo arrivo ad Alessandria, appunto per educare il figlio di Tolomeo III, portò in dono al re un componimento poetico, sulla dimostrazione matematica del problema della duplicazione del cubo, e un modello in bronzo dello strumento ideato da lui per risolvere il problema, il *mesolabio*, dal greco mesølaboß che significa *atto a prendere nel mezzo*. Oggi, Eratostene viene ricordato essenzialmente per due grandi contributi: un metodo per distinguere i numeri primi dai numeri composti, conosciuto come *crivello di Eratostene*; e la misura del meridiano terrestre.

II *crivello* è un setaccio, come quello, per esempio, che serve per separare la crusca dalla farina, ed il nome è scelto appropriatamente. Infatti, basta scrivere i numeri naturali in successione in una tabella, escludendo il numero *1*, e come primo passo togliere tutti i multipli di *2:*

2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41
42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61
62	63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80	81
82	83	84	85	86	87	88	89	90	91
92	93	94	95	96	97	98	99	100	...

2	3		5		7		9		11
	13		15		17		19		21
	23		25		27		29		31
	33		35		37		39		41
	43		45		47		49		51
	53		55		57		59		61
	63		65		67		69		71
	73		75		77		79		81
	83		85		87		89		91
	93		95		97		99		...

Nel secondo passo si cancellano tutti i multipli di *3*, e nel terzo tutti i multipli di 5, poi di 7, e così via:

2	3		5		7				11
	13				17		19		
	23		25				29		31
			35		37				41
	43				47		49		
	53		55				59		61
			65		67				71
	73				77		79		
	83		85				89		91
			95		97				...

2	3		5		7				11
	13				17		19		
	23						29		31
					37				41
	43				47				
	53						59		61
					67				71
	73						79		
	83						89		
					97				...

Ciò che rimane è appunto l'insieme dei *numeri primi*:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, . . .

È incredibile come questo semplice ed elegante procedimento per determinare i numeri primi sia alla base di alcune generalizzazioni che hanno permesso di calcolare tavole complete di numeri primi fino a circa *10 000 000*, che forniscono una quantità enorme di dati empirici sulla distribuzione di questi numeri misteriosi. Proprio sulla base di queste tavole si sono formulare e si formulano molte congetture sui numeri primi, che spesso sono molto semplici da enunciare ma difficilissime da dimostrare.
Una delle congetture più famose è quella di Goldbach, ancora irrisolta, secondo la quale *ogni intero pari maggiore o uguale a 6 si può sempre scrivere come la somma di due primi dispari*.

Le tavole dei numeri primi riempivano, un tempo, molte pagine, per cui ci si può chiedere in che modo esse vengano conservate in un computer. Ebbene, uno dei metodi più semplici per fare ciò è quello di codificare ciascun numero dispari mediante le cifre binarie, facendo corrispondere ad un numero la cifra 0, se esso non è primo, e la cifra 1 se invece è primo. In questo modo, si possono rappresentare i numeri dispari a blocchi di cinque, da 1 a 9, da 11 a 19, ecc., mediante numeri in base due di cinque cifre. Così, per esempio, il numero 01110 significherà che il numero 1 non è primo, 3 è primo, 5 è primo, 7 è primo 9 non lo è.

0 1 1 1 0	1 1 0 1 1	0 1 0 0 1	1 0 0 1 0
1 3 5 7 9	11 13 15 17 19	21 23 25 27 29	31 33 35 37 39

d) Costruzione di un algoritmo per contare i numeri primi minori di un numero assegnato

Possiamo utilizzare il metodo di Eratostene proprio per contare i numeri primi non superiori ad un certo numero naturale prefissato *n*, cioè per determinare uno o più valori della funzione $\pi(n)$.
Poiché tutti i numeri primi, eccetto *2*, sono dispari, ad ogni numero dispari $n > 3$ possiamo associare una *variabile booleana* che assumerà il valore «vero» se il numero in questione è primo, e il valore «falso» se esso non è primo. Avremo bisogno dunque di usare un *vettore* di variabili booleane (cioè, quello che nel linguaggio *Turbo-Pascal* viene denominato come un *array*, che è un *dato strutturato*, in quanto composto da più elementi), che contenga almeno $\frac{n-1}{2}$ elementi. Inizialmente tutti gli elementi di questo vettore vengono posti uguali a «vero», come se fossero tutti primi; successivamente, gli elementi del vettore vengono scanditi, partendo dal primo elemento che corrisponde al numero *3*, e ogni volta che si incontra un elemento il cui valore è «vero» (cioè, corrispondente ad un numero primo) si pongono uguale a «falso» tutti gli elementi del vettore che corrispondono ai multipli del numero primo trovato. Inoltre, è necessario contare i numeri primi trovati, per cui è necessario usare un *contatore*, che inizialmente si pone uguale ad *1*, per tenere conto del fatto che il numero *2* non viene determinato dal procedimento proposto.
Se chiamiamo con *F* il nostro vettore, allora *F(1)* corrisponde a *3*, *F(2)* corrisponde a *5*, e così via; cioè, in generale, *F(I)* corrisponderà al numero dispari $P = 2 \cdot I + 1$. Appena si sarà individuato un indice *I* tale che il numero dispari $2 \cdot I + 1$ sia primo (quindi, *F(I)* «vero»), sarà sufficiente porre uguale a «falso» tutti i multipli dispari di *P* (cioè tutti i numeri del tipo $P \cdot P$, $(P+2) \cdot P$, $(P+4) \cdot P$, ...

L'algoritmo avrà la struttura seguente;

```
0.      N : = n
1.      C : = 1, KMAX : = (N - 1)/2
2.      per I = 1, 2, ... , KMAX, ripetere:
2.1             F(I) : = «vero»
3.      per I = 1, 2, ... , KMAX, ripetere:
3.1             se F(I) = «vero»,
                allora:
3.1.1                   P : = 2 · I + 1
```

3.1. 2K : = (P · P - 1) div 2
3.1.3 finché K ≤ KMAX, ripetere:
3.1.3.1 F(K) : = «falso»
3.1.3.2 K : = K + P
3.1.4 C : = C + 1
4. stampare C
5. fine.

e] Costruzione di un algoritmo per decidere se un numero sia primo o non primo.

Costruiamo un algoritmo tale che, per ogni dato numero naturale *N*, ci consenta di stabilire se *N* sia primo o non primo, e in quest'ultimo caso, quali siano i suoi divisori non banali, cioè diversi da 1 e da N stesso.
È chiaro che, se N è uguale a 2, l'algoritmo ci deve dare il messaggio: "*N* è primo". Se invece N è maggiore di 2 consideriamo un'altra variabile, diciamo *X*, alla quale assegneremo, successivamente, e uno alla volta, tutti i numeri interi che vanno da *2* fino ad *N - 1*, poi eseguiamo le divisioni del tipo N=X·Q+R dove *Q* ed *R* sono il quoziente e il resto della divisione.
Esaminando i valori della variabile *R* si riuscirà a stabilire se *N* è primo oppure non è primo. Infatti:
- se *R = 0* per almeno un valore di *X*, allora tale valore è un divisore di *N* per cui 'N non è primo'.
- se *R ≠ 0* sempre per tutti i valori di *X*, allora nessuno di tali valori è divisore di *N*, per cui '*N* è primo'.
Per costruire l'algoritmo introduciamo un *indicatore* (o *spia*), detto *flag* in inglese, che possa trovarsi solo in due *stati*:

1° stato: spia = 0 (cioè, *spia spenta*)

2° stato: spia = 1 (cioè, *spia accesa*)

Questa spia ci consentirà di avere informazioni relative ad un certo evento, che nel nostro caso sarà quello d'avere trovato, tra i valori di *X*, variabili tra *2* e *N - 1*, un divisore di *N* oppure no, secondo lo schema:

SE spia = 0
ALLORA l'evento non si è verificato
ALTRIMENTI, se spia = 1, l'evento si è verificato.

Allora, si farà in modo che la *spia* sia inizialmente spenta, poi, solo se tra i valori di *X* si trova un divisore di *N*, allora si assegnerà alla spia il valore 1, cioè essa si accenderà; in caso contrario, essa rimarrà sempre spenta.
Dunque, nel nostro problema, l'evento è *la scoperta di un divisore di N*; se questo evento si verifica, allora si potrà affermare che '*N* non è primo'; se invece non si verifica, allora si dedurrà che '*N* è primo', e si visualizeranno tutti i divisori non banali di *N*.

```
Inizio
introduci N;
se (N = 2)
allora visualizza ('N è primo')
altrimenti
inizio
spia : = 0;
X : = 2;
Ripeti
Q : = N div X; (quoziente intero)
R : = N - Q · X;
se R = 0 allora

inizio
visualizza ('X è divisore di N');
spia : = 1
fine;
X := X + 1

finché X ≥ N - 1;
se spia = 0
allora visualizza ('N è primo')
altrimenti visualizza ('N non è primo');
fine;
fine.
```

f] Costruzione di un algoritmo per scomporre un numero in fattori primi

Quando scomponiamo un numero in fattori primi, come per esempio 60, il metodo che seguiamo può essere descritto nel modo seguente.
Cominciamo a provare se esso sia divisibile per il primo numero primo, cioè per 2; se lo è eseguiamo la divisione e scriviamo il quoziente. Riproviamo se il quoziente ottenuto sia ancora divisibile per 2; se lo è calcoliamo il nuovo quoziente, oppure proviamo se quest'ultimo è divisibile per il numero primo successivo di 2, cioè 3. Se la divisione per 3 dà un nuovo quoziente, riproviamo per esso la divisibilità per 3. Se non va bene, proviamo se il nuovo quoziente è divisibile per il numero primo che viene dopo 3, cioè 5. Se esso è divisibile per 5, scriviamo il nuovo quoziente, e su questo ripetiamo nuovamente lo stesso tipo di ragionamento, finché non otterremo come quoziente 1. A questo punto la scomposizione sarà terminata. Tutto ciò potrà essere sintetizzato nella tabella seguente.

60	2	si prova se 60 è divisibile per 2 si fa il quoziente e si riprova se	si
30	2	è divisibile per 2 si fa il quoziente e si riprova se è divisibile ancora per 2	si no
15	3	si prova se è divisibile per 3 si fa il quoziente si riprova se è divisibile per 3	si no
5	5	si prova se è divisibile per 5 si fa il quoziente	si
1		il quoziente è 1	fine

La logica dell'algoritmo dovrà, quindi, ricalcare in parte il modo che abbiamo esaminato prima, per cui può essere descritto così:

```
1.      Inizio
2.          Scrivi 'introduci il numero da scomporre'
3.          Leggi n
4.          i ← 2
4.1             Finché n ≠ 1 esegui
4.1.1               Inizio
4.1.2                   Finché n mod i ≠ 0 esegui
4.1.3                       i ← i + 1
4.1.4                   fine
5.                  scrivi i
6.                  n ← n div i
7.              fine
8.      Fine.
```

Altri algoritmi del passato

Hai pensato a un nome da dargli? Che ne dici di "La canzone di Pitagora" ?

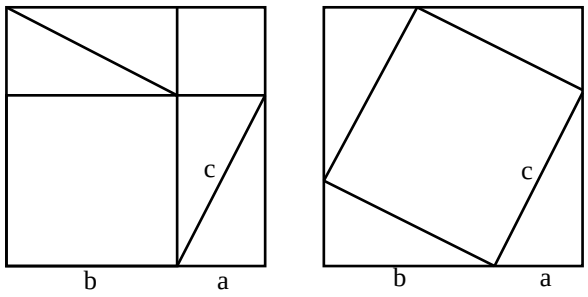
D.R. Hofstadter, *Gödel, Escher, Bach*.

Domandò il Coniglio Bianco: "Da dove devo incominciare, Maestà?". "Incomincia dal principio", disse molto gravemente il Re, "e vai avanti fino alla fine: allora fermati".

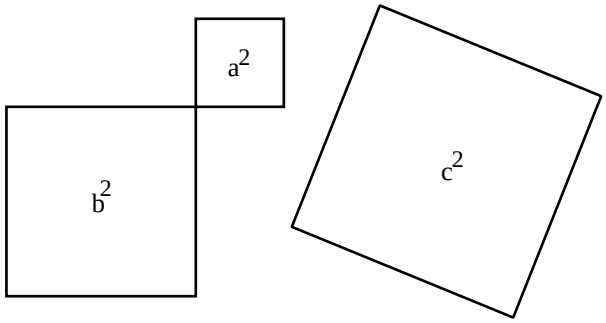
L. Carroll, *Alice nel Paese delle meraviglie.*

1] Algoritmo babilonese per la ricerca delle radici, in N, di un'equazione di secondo grado.

Come è noto, nell'antichità, la dimostrazione matematica era soprattutto una rappresentazione grafica, e molte delle regole usuali del nostro moderno simbolismo matematico hanno avuto in precedenza delle rappresentazioni a *mosaico*. Così, per esempio, i Babilonesi, già molto prima del 2000 a.C., usarono questo tipo di rappresentazioni per mettere a punto molti problemi. Un esempio famoso di *dimostrazione grafica* è quella del "teorema di Pitagora".
Bastava infatti osservare le due figure:



e dal loro confronto dedurre che $c^2=a^2+b^2$. Infatti, ciascun quadrato ha il lato di lunghezza $a+b$. Il primo quadrato è diviso in sei parti, un quadrato di lato a, uno di lato b e quattro triangoli rettangoli di lati a, b e c. Il secondo quadrato è diviso in cinque parti: un quadrato di lato c e quattro triangoli rettangoli di lati a, b e c. Sottraendo dai due quadrati i quattro triangoli rettangoli rimangono i quadrati



che risultano uguali perché ai due quadrati di lato $a+b$ è stato sottratto lo stesso numero di parti uguali.

Tornando ai Babilonesi, in una tavoletta di argilla, giunta fino a noi, è descritto un metodo per risolvere l'equazione di secondo grado del tipo:

$$x^2 - M \cdot x + N = 0$$

Per vedere se essa ha radici in *N*, si costruisce una tavola in cui compare la tabulazione dell'espressione relativa a diversi valori di *N*, per diversi valori di *x* (precisamente per *M* valori di *x*). Se il numero che compare ad un incrocio è uguale al prodotto *M·x*, allora la *x* corrispondente è soluzione dell'equazione.
Per esempio, usando la tabella che segue, si trova che l'equazione $x^2 - 5 \cdot x + 6 = 0$ è soddisfatta per $x=2$ e per $x=3$.

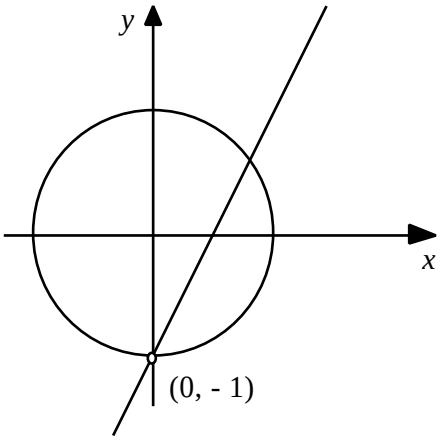
x \ N	1	2	3	4	5	6
1	2	3	4	5	6	7
2	5	6	7	8	9	10
3	10	11	12	13	14	15
4	17	18	19	20	21	22
5	26	27	28	29	30	31

2] Algoritmo per determinare una tavola di terne pitagoriche

Si chiama *pitagorica* una terna di numeri interi positivi che possano essere le misure dei lati di un triangolo rettangolo, e che quindi soddisfano la relazione

$$a^2 = b^2 + c^2$$

È immediato osservare che se *a, b, c* formano una terna pitagorica, altrettanto accade per *ka*, *kb*, *kc*, con *k* numero naturale. Il problema di cercare terne pitagoriche primitive si riduce, quindi, a quello di determinare *a, b, c* in modo che siano primi tra loro. Ne segue intanto che *a* e *b* non possono essere entrambi pari, perché, altrimenti, lo sarebbe anche *c*, e la terna non sarebbe primitiva. Né possono essere entrambi dispari, perché la somma di due quadrati dispari non è un quadrato. Si può supporre, quindi, *b* pari ed *a* e *c* dispari.
La relazione $a^2 = b^2 + c^2$ possiamo anche scriverla nella forma $x^2 + y^2 = 1$, con $x = \frac{a}{c}$ e $y = \frac{b}{c}$
.
Affinché *a, b e c* siano interi, *x* e *y* devono essere razionali, e poiché l'equazione $x^2 + y^2 = 1$ rappresenta una circonferenza di centro l'origine degli assi e raggio *1*, il nostro problema si riconduce a quello di determinare i punti di coordinate razionali appartenenti a questa circonferenza.
A tale scopo consideriamo una retta passante per il punto $(0, -1)$ di equazione $y = m\,x - 1$ e determiniamone i punti di intersezione con la circonferenza.



Questi punti avranno come coordinate:

$$x = \frac{2m}{1 + m^2} \qquad y = \frac{m^2 - 1}{1 + m^2}$$

Queste equazioni forniscono una rappresentazione parametrica della circonferenza senza l'uso di funzioni goniometrie.
Poiché nel nostro problema ci interessano solo i valori razionali positivi di (x, y), allora i valori di m dovranno essere maggiori di 1 e razionali. Ponendo nelle formule precedenti $m = \frac{i}{k}$ (con i e k naturali dispari e primi tra loro, e $i > k$) e dividendo per 2 si ottiene:

$$x = \frac{k\,i}{\frac{(k^2 + i^2)}{2}} \qquad y = \frac{\frac{(i^2 - k^2)}{2}}{\frac{(k^2 + i^2)}{2}}$$

Ricordando le posizioni fatte, avremo

$$a = k\,i \qquad b = \frac{(i^2 - k^2)}{2} \qquad c = \frac{(k^2 + i^2)}{2}$$

per cui, ad ogni coppia k, i che soddisfa le condizioni richieste, corrisponderà una terna pitagorica primitiva.
Per produrre un algoritmo che calcoli le terne pitagoriche primitive si deve introdurre un naturale dispari da assegnare alla variabile *i*, e mediante due cicli prendere in considerazione tutte le coppie di naturali dispari comprese tra *i* e il numero. Inoltre, si può utilizzare una *function* che calcoli il m.c.d. tra *k* ed *i*, scartando le coppie di numeri non primi tra loro.
L'algoritmo, scritto in Pascal è il seguente:

```
Program ternepit;
var n, i, k, x, y, z: integer;
function mcd (a, b: integer) : integer;
var resto : integer;
        begin
          resto: = a mod b;
            while resto <>0 do
```

```
            begin
               a: = b;
               b: = resto;
               resto: = a mod b
         end;
      mcd: = b
   end;

begin
      writeln ('introduci n dispari');
      writeln;
      read (n);
      writeln;
      i: = n;
      while i>=3 do
          begin
             k: = i - 2;
             while k >= 1 do
                  begin
                      if mcd (k, i) = 1 then
                         begin
                              x:= i*k;
                              y:= (i*i - k*k) div 2;
                              z:= (i*i + k*k) div 2;
                              write (x:7,   '       |',y:7,'          |',z:7,'         | |')
                         end;
                      k:= k-2
              end;
                i:=i-2
      end
end.
```

3] Algoritmo greco per il calcolo della radice quadrata di un numero

Questo algoritmo può essere considerato un metodo di calcolo per tentativi, e si fonda su opportune considerazioni sul concetto di media aritmetica e di media geometrica tra due numeri.
Mediante questo metodo il valore della radice quadrata di un numero viene calcolato mediante successive approssimazioni a partire da una stima iniziale della radice.
Illustriamolo con un esempio numerico. Supponiamo di voler calcolare la $\sqrt{60}$. Si possono elencare i seguenti passi:

Passo 1. Si fornisce una stima della radice, ad es. 6: $u_1 = 6$.

Passo 2. Si divide 60 per u_1: $\frac{60}{6} = 10$.

Poiché la media geometrica fra 6 e *10* è $\sqrt{6 \ 10}$ si deduce che $\sqrt{60}$ è un numero compreso tra 6 e *10*.

Passo 3. Si fa la media aritmetica fra u_1 e $\frac{60}{u_1}$: $u_2 = \frac{u_1 + \frac{60}{u_1}}{2}$

Nel nostro caso $u_2 = 8$; per cui u_2 è ovviamente, una seconda approssimazione di $\sqrt{60}$ rispetto alla stima iniziale.

Passo 4. Si ripetono i passi a partire dal passo 2. fino al numero di iterazioni che si desiderano. Nel nostro esempio:

$$\frac{60}{8} = 7,5$$

$$\frac{8 + 7,5}{2} = 7,75.$$

Proseguendo si ha:

$$u_3 = \frac{u_2 + \frac{60}{u_2}}{2}$$

.....

$$u_n = \frac{u_{n-1} + \frac{60}{u_{n-1}}}{2}$$

L'algoritmo, scritto in Pascal, è il seguente:

```
program radgreca;

var n, num, i, rad: integer;
u: array[1..100] of real;

begin
        writeln (‘Introduci il n.ro di cui vuoi la radice’);
        read (num);
        writeln;
        writeln (‘Introduci il numero di iterazioni’ [ < 100]’);
        read (n);
        writeln;
        writeln (‘Introduci la prima stima della radice’);
        read (rad);
        u[1]: = rad;
                for i:=2 to n do
                        u[i]:=1/2*(u[i-1]+ num/u[i-1]);
        writeln;
        writeln (‘La radice di’, num,’ e’’’, u[i])
end.
```

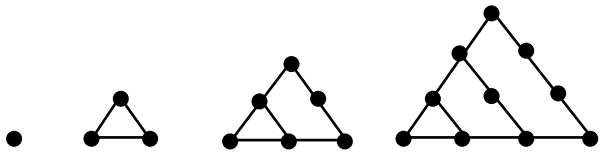
4] *Un algoritmo di Nicomaco*

Nel 100 d.C. Nicomaco scoprì la possibilità di esprimere il cubo di un numero naturale *n* come somma di *n* numeri dispari. Gli *n* termini di tale somma differiscono tra loro di *2*.
Per esempio:

$$1^3 = 1$$
$$2^3 = 3 + 5$$
$$3^3 = 7 + 9 + 11$$
$$4^3 = 13 + 15 + 17 + 19$$
$$5^3 = 21 + 23 + 25 + 27 + 29$$

L'algoritmo che permette di determinare i termini delle somme di dispari che rappresentano i cubi dei naturali è fondato sulle seguenti osservazioni:

a) Se è possibile determinare l'ultimo termine della somma, allora sono noti tutti i termini della somma.
b) L'ultimo termine della somma è un numero dispari il cui ordine è dato dall'ennesimo numero triangolare. Ciò vuol dire che, per esempio, nella scomposizione $2^3 = 3 + 5$, il 5 è il *terzo* numero dispari, e il suo ordine (*terzo*) è uguale al numero triangolare *3* che è il *secondo* numero triangolare. Insomma, la base del cubo mi fornisce l'indicazione sull'ordine di un numero triangolare, e il valore di quest'ultimo mi fornisce il posto occupato dal numero dispari cercato nella successione dei numeri dispari.
Se gli allievi dovessero essere meravigliati dalla dizione “numero triangolare”, si avrebbe una ottima occasione per illustrare loro un po' della matematica pitagorica, e parlare così dei numeri figurati, limitandosi magari alle configurazioni piane. Si potrebbe dire, per esempio, che i Pitagorici classificavano i numeri in triangolari, quadrati, poligonali, cubici, oblunghi e così via secondo le forme in cui erano disposti i punti.
Triangolari erano detti i numeri per i quali i punti assumevano la forma di un triangolo, come 1, 3, 6, 10, ...



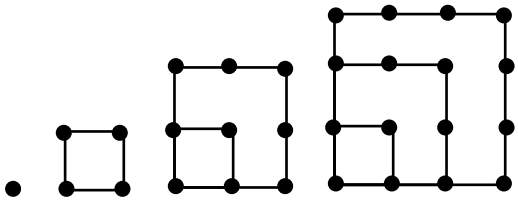
Da queste disposizioni geometriche essi deducevano che i numeri triangolari si ottenevano sommando successivamente i numeri naturali. Infatti:

$$1+2=3$$
$$1+2+3=6$$
$$1+2+3+4=10$$
$$1+2+3+4+5=15$$
$$1+2+3+4+5+6=21$$
.....

Per cui, utilizzando il nostro simbolismo, l'*n*-esimo numero triangolare è dato da:

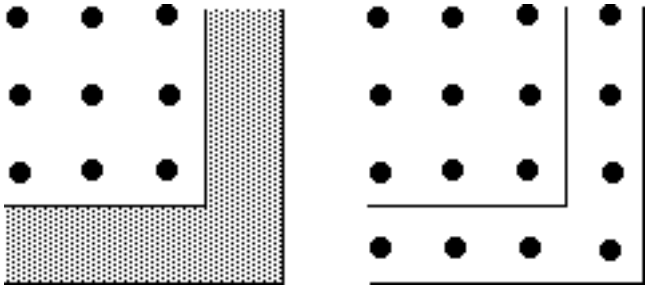
$$1+2+3+4+...+ n = n\frac{(n+1)}{2}$$

Quadrati erano detti tutti i numeri che si rappresentavano disponendo i punti in modo da formare un quadrato, come *1, 4, 9, 16* e così via:



1 4 9 16

Per passare da un numero quadrato al successivo, i Pitagorici utilizzavano un metodo grafico che consisteva in ciò: tracciavano due "lati" del quadrato di partenza e aggiungevano tanti punti quanti erano necessari per formare un altro quadrato:



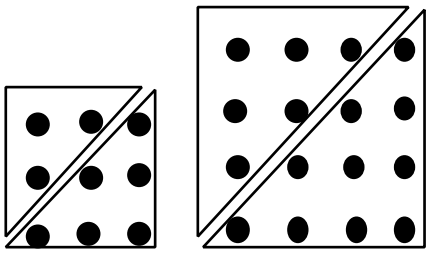
In realtà, essi non facevano altro che verificare una proprietà algebrica elementare che oggi può essere espressa nel modo seguente:

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

Inoltre, partendo da 1 e aggiungedovi lo gnomone formato da tre punti, e a quello ottenuto lo gnomone formato da 5 punti, e così via si potevano ottenere i vari numeri quadrati mediante la somma dei successivi numeri dispari, che noi oggi esprimiamo così:

$$1 + 3 + 5 + 7 + ... + (2n - 1) = n^2$$

Osservando la forma dei numeri quadrati i Pitagorici dedussero che un numero quadrato poteva essere ottenuto anche dalla somma di due numeri triangolari successivi:

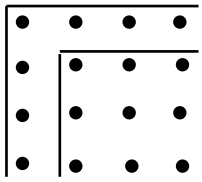


9 = 3 + 6 16 = 10 + 6

Questo, in generale, è vero perché:

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

Inoltre, questa configurazione permise ai pitagorici di scoprire che ogni numero dispari è la differenza di due quadrati; per esempio:



$$7 = 4^2 - 3^2$$

Si potrebbe continuare ancora con i numeri *pentagonali*, quelli *esagonali*, e così via. Anzi, si potrebbe suggerire agli allievi di determinare la loro forma generale.

Tornando all'algoritmo, possiamo stabilire i passi seguenti:

Passo 1. Si calcola innanzitutto l'*n*-esimo numero triangolare dato dalla formula:

$$N_n = n \frac{(n+1)}{2}$$

Passo 2. Poiché il *k*-esimo numero dispari è dato da (*2k-1*), sostituendo la formula dell'ennesimo numero triangolare al posto di *k*, si ottiene l'ultimo termine degli *n* numeri dispari la cui somma è il cubo di *n*:

$$\text{ultimo termine} = 2 \cdot \left[\frac{n \cdot (n+1)}{2} \right] - 1 = n^2 + n - 1$$

Se rappresentiamo la somma con:

$$S = (x + 1) + (x + 3) + \ldots + (x + 2n - 1)$$

ne segue che l'ultimo termine è dato da $n^2 + n - 1$ se e solo se

$$x = n^2 - n.$$

L'algoritmo in Pascal, in cui si usa un array per esprimere la sequenza di numeri dispari la cui somma è il cubo di *n*, é il seguente:

```
program Nicomaco;
var     k, n x: integer;
        sequenza: array [1..1000] of integer;
        triangolare: array [1..1000] of integer;

begin
        writeln (' introduci n');
        read (n);
        writeln; writeln;
                for k:=1 to n do
                        triangolare[k]: = (k-1) *2 + 1;
                        writeln;
                        x : = n * n - n;
                                for k : = 1 to n do
                                        sequenza[triangolare [k]]:=triangolare[k]+x;
                        writeln (' Il cubo di', n, ' è la somma di questi numeri:');
                        writeln;
                                for k: = 1 to n do
                                        write (sequenza[triangolare[k]],' ')
end.
```

Bibliografia essenziale

1] G.C. Barozzi, *Algoritmi sui numeri interi*, Quaderni del CNR, 1986.

2] Jean-Luc Chabert et al., *A History of Algorithms*, Springer, 1999.

3] L. Childs, *A Concrete Introduction to Higher Algebra*, Springer, 1978.

4] R. Courant & H. Robbins, *Che cosé la Matematica?* (II ed. a cura di I. Stewart), Universale Bollati Boringhieri, 2000.

5] Donald E. Knuth, *The Art of Computer programming*, Addison-Wesley, v. 1, Third edition, 1997.

6] G. Lolli & C. Mangione (ed.), *Matematica e Calcolatore*, Le Scienze, 1984.

7] F. Luccio, *La struttura degli algoritmi*, Boringhieri, 1982.

8] F. Luccio-E. Lodi-L. Pagli, *Algoritmica*, Quaderni del CNR, 1986.

9] A. Machì, *Algebra per il Calcolo Simbolico*, Edizioni Kappa, 1995.

10] C.D. Olds, *Frazioni continue*, Zanichelli, 1970.

11] O. Ore, *Number Theory and his History*, Dover, 1988.

12] R. Penrose, *La mente nuova dell'Imperatore*, Rizzoli, 1992.

13] J. Roberts, *Elementary Number Theory*, The MIT Press, 1978.

14] R. Sedgewick, *Algorithms*, Addison-Wesley, 1983.

15] P. Zellini, *Gnomon*, Adelphi, 1999.

Indice