algóre, m. *ALGOR- ŌRIS. Freddo grande. | Stagione fredda.

+algorişmo, -itmo, m. *sp. ALGUARISMO cifra arabica (dal matematico ar. Al-Kuarismi che apprese dagl'Indiani l'abbaco e l'insegnò in Spagna circa l'820). Aritmetica col sistema arabo. | Pratica dell'aritmetica.

algoso, v. sotto algo. || **+alia,** v. ala.

ali are, nt. ALIA. Muover l'ali, Svolazzare, Aleggiare. Volare. | Aggirarsi, degl' innamorati, Ronzare. || **+-eggiare,** nt., frq. Aliare; Andare errando.

àlias, av. l. Altrimenti, Con altro nome. | In altro tempo, Già, Ex, Quondamo.

àlibi, (l. Altrove). m. ⚖ Dichiarazione di essersi trovato in luogo diverso da quello in cui fu commesso il delitto.

alicante, m. Squisito vino spagnolo, dal luogo d'origine (Valencia).

alic e, f. *HALEC -ĒCIS. ⇦ Acciuga. || **-etta,** m. dm. | Argentina.

alicòrno, m. Liocorno, animale favoloso. | edicamento fatto col dente o l corno dell'unicorno.

ìcula, f. *ALĬCŬLA. 🎧 Tunica corta avvilupte le spalle usata dai Romani.

dada, f. *ar. AL IDADA. Regolo mobile rnato nel centro d'uno stru- fatto per misurare gli : Dioptra, Traguardo.

Alicula.

presa, ecc. | pl. ⚖ vestire e alloggio, e di legge o testamento al nutrimento. | *gene* alimento. | Mante gli alimenti. || **-amer** tare. || **-ario,** ag. * mento o cibo. | *cond* cibi. | m. ⚖ Colui a || **-atore,** m. **-atrice,** muove e fornisce. || - Nutrizione. | Alimen Alimentario. || **-oso,**

+alim o, m. (*a* non h asfodelo o altro, ch la fame. | (*àlimo*). * || **-urgìa,** f. *ἔργον alimentari.

+alimònia, f c per sua colpa dal

alinea, m. *A poverse

+aliòsso, m. di vano i ragazzi. Ta

aliòtico, m. 🎧 Basti

aliòtide, f. *ἅλιος orecchie. ⇦ Orecc

aliòtto, m., di della so la spalla.

La nozione di algoritmo (vedi anche [SA15, Cap. 8])

Eugenio G. Omodeo
- Università di Trieste -

Trieste, 21.10.2020

Computabilità = Teoria delle funzioni ricorsive

Certo, se la funzione ricorsiva generale è l'equivalente formale della computabilità effettiva, la sua formulazione può svolgere un ruolo nella storia della matematica combinatoria secondo solo a quello della formulazione del concetto di numero naturale.

(Emil L. Post, 1944)

(*Emil Leon Post*, 1897–1954)

Etimologia del vocabolo "**algoritmo**"

L'origine di questa parola si ricollega al nome del grande matematico arabo Muḥammad *ibn*-Mūsà vissuto alla corte di Baghdad nei primi decenni del secolo IX. Nativo di Hwarizm, regione dell'Asia centrale, egli fu soprattutto noto come *al*-Huwarīzmī, nome latinizzato poi in Algorismus. Il termine finí per significare ogni *procedimento di calcolo puramente schematico*.

Esempi basilari

Oltre ai procedimenti per effettuare le operazioni di somma, sottrazione, moltiplicazione, divisione, innumerevoli altri ne vengono escogitati di continuo,

sia NUMERICI che SIMBOLICI

Qualche esempio:

- calcolo del resto di una divisione fra numeri interi
- calcolo del massimo comun divisore e minimo comune multiplo
- estrazione della radice quadrata (o cubica) di un numero
- individuazione dell'N-esimo numero primo
- determinazione dell'N-esima cifra di π
- confronto alfabetico fra due parole
- operazioni su polinomi

Un esempio classico: l'**algoritmo di Euclide** – I

Problema: Cerchiamo il massimo comun divisore di due numeri (naturali, di cui almeno uno diverso da 0).

Procedimento risolutivo: Indichiamo con X ed Y i due operandi

- Se $Y = 0$, il M.C.D. cercato è X ;
- altrimenti si calcoli il resto della divisione di X per Y, ossia quel numero r tale che per un opportuno q (detto quoziente), valga che

$$X = q \cdot Y + r \quad \text{con } 0 \leqslant r < Y$$

- Riduciamo il problema originario a quello di cercare il M.C.D. tra Y ed r

Un esempio classico: l'**algoritmo di Euclide** – II

Esempio: Prendiamo come dati d'ingresso: $X = 132$ ed $Y = 252$

X	Y	
132	252	
252	132	
132	120	
120	12	
12	0	$\surd$

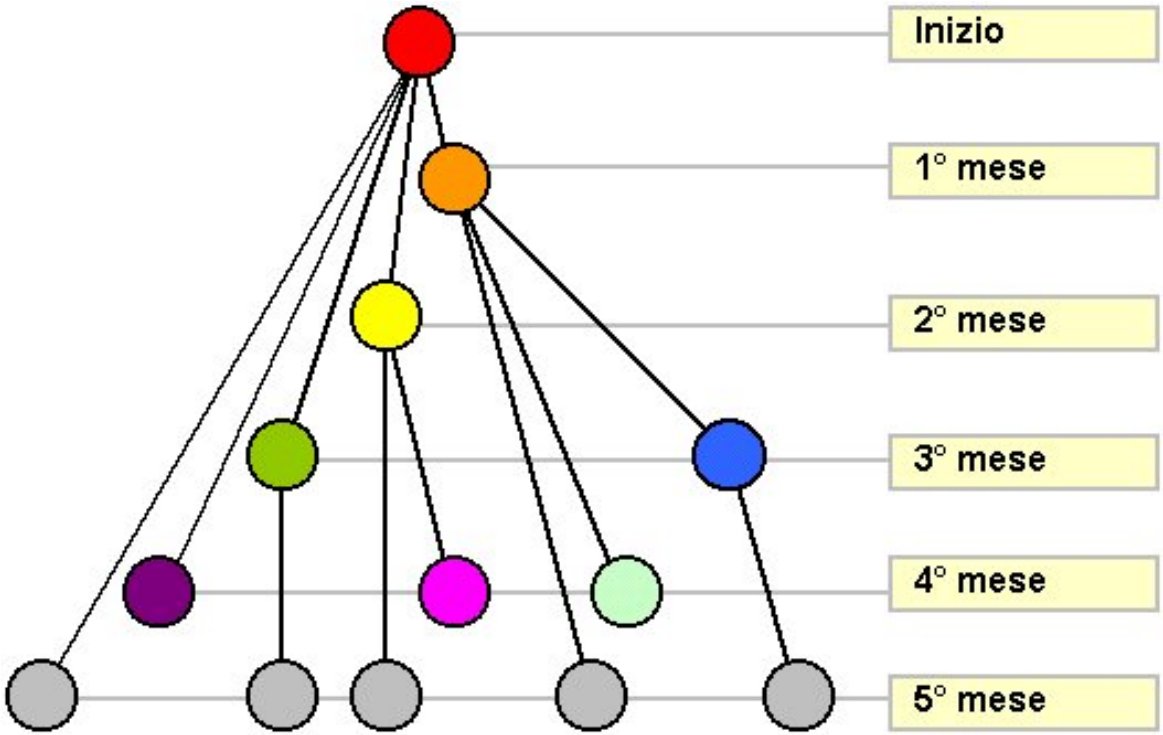
Altro esempio storico: **progressione di Fibonacci – I**

Il "*maestro d'abaco*" Leonardo Fibonacci (Pisa, 1175–1235 ca.) si pose il seguente

Problema: Un tale mise una coppia di conigli in un luogo completamente circondato da un muro, per scoprire quante coppie di conigli discendessero da questa in un anno: per natura le coppie di conigli generano ogni mese un'altra coppia e cominciano a procreare a partire dal secondo mese dalla nascita.

Vedi http://utenti.quipo.it/base5/fibonacci/fibonacci.htm

Altro esempio storico: **progressione di Fibonacci** – II



Progressione: (1); 1, 2, 3, 5, 8, 13; 21, 34, 55, 89, 144, 233, 377;

. . .

Altro esempio storico: **progressione di Fibonacci** – III

Procedimento risolutivo: Indichiamo con N il numero dei mesi (per es. $N = 12$)

- Se $N = 0$ oppure $N = 1$, il numero di coppie è 1 ;
- altrimenti, riduciamo il problema a *due* problemi analoghi, riguardanti $N - 1$ ed $N - 2$ rispettivamente;
- una volta risolti separatamente i due sotto-problemi, si sommino i due risultati per ottenere la soluzione del problema originario

Analisi critica dei due procedimenti appena visti

Si tratta di *circoli viziosi*? Piuttosto, di strategia *divide et impera*!

> *La regola dell'attacco pianificato in guerra è: se le nostre forze sono dieci volte quelle del nemico, basterà circondarlo; se cinque a uno, occorrerà attaccarlo; se il numero sarà il doppio, occorrerà dividere l'armata del nemico in due.*
>
> **Sun Tzu, VI sec. a.C., "L'arte della guerra"**

La teoria che serve da sfondo allo studio degli algoritmi, viene spesso chiamata TEORIA DELLA RICORSIONE

Software e programmazione

I linguaggi in cui si realizza il software possono essere

- Turing-completi
- di nicchia, e.g.:
  - *grammatiche*, automi
  - linguaggi per *basi di dati*
  - linguaggi di *markup*

Software e programmazione

I linguaggi in cui si realizza il software possono essere

- Turing-completi
- di nicchia, e.g.:
  - *grammatiche*, automi
  - linguaggi per *basi di dati*
  - linguaggi di *markup*

☞: un linguaggio *completo*, benché ridotto all'essenziale

Software e programmazione

I linguaggi in cui si realizza il software possono essere

- Turing-completi
- di nicchia, e.g.:
  - *grammatiche*, automi
  - linguaggi per *basi di dati*
  - linguaggi di *markup*

Altra classificazione:

- linguaggi di *rapida prototipazione*
- linguaggi di *produzione*
  - e.g., orientati agli oggetti

Riferimenti bibliografici


Lawrence Snyder and Alessandro Amoroso.
FLUENCY –Conoscere e usare l'informatica.
Pearson Italia, Milano-Torino, 5ª edition, 2015.

- Per semplicità, supponiamo che la base b soddisfi

$$1 < b \leqslant 36\,.$$

- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Per semplicità, supponiamo che la base b soddisfi

$$1 < b \leqslant 36\,.$$

- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""

- Per semplicità, supponiamo che la base b soddisfi

$$1 < b \leqslant 36\,.$$

- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""
- Poi, **ripetutamente,**

- Per semplicità, supponiamo che la base b soddisfi
$$1 < b \leqslant 36\,.$$
- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""
- Poi, **ripetutamente,**
  - calcolare `quoz` $= N/b$;

- Per semplicità, supponiamo che la base b soddisfi
$$1 < b \leqslant 36\,.$$
- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""
- Poi, **ripetutamente,**
  - calcolare `quoz` $= N/b\,;$ `res` $= N - (N/b)\cdot b\,;$

APPENDICE: Un algoritmo per rappresentare un intero $N \geqslant 0$ in base b

- Per semplicità, supponiamo che la base b soddisfi
$$1 < b \leqslant 36\,.$$
- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""
- Poi, **ripetutamente,**
  - calcolare `quoz` $= N/b$; `res` $= N - (N/b) \cdot b$;
  - inserire `cifre[res]` all'inizio di `numerale`;

- Per semplicità, supponiamo che la base b soddisfi
$$1 < b \leqslant 36\,.$$
- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""
- Poi, **ripetutamente,**
  - calcolare `quoz` $= N/b$; `res` $= N - (N/b)\cdot b$;
  - inserire `cifre[res]` all'inizio di `numerale`;

  fintantoché $N \neq 0$.

- Per semplicità, supponiamo che la base b soddisfi
$$1 < b \leqslant 36\,.$$
- Per rappresentare le `cifre`, ci serviremo di questa sequenza di caratteri:

'0'	'1'	···	'9'	'a'	'b'	···	'z'
0	1	...	9	10	11	...	35

- Cominciare ponendo `numerale` $=$ ""
- Poi, **ripetutamente,**
  - calcolare `quoz` $= N/b$; `res` $= N - (N/b)\cdot b$;
  - inserire `cifre[res]` all'inizio di `numerale`;

  fintantoché $N \neq 0$.
- Risultato: La seq. di caratteri assemblata in `numerale`.