Introduzione agli algoritmi e alle strutture di dati

" le parole sono importanti... "

Esposizione degli elementi essenziali, concetti e nozioni di base, di una disciplina. Con un linguaggio semplice.

Modalità di organizzazione ed aggregazione dei dati utili alla messa a punto di metodi risolutivi per problemi di interesse...

... ci torneremo...

Algoritmi

Forma latina medievale "Algorismus"
(presente in un testo del 1330)

Deriva dal nome del matematico, astronomo e geografo persiano Abū Ja Far Muhammad ibn Musa al-Khwarizmi (780 - 850 d.C)

"Visse a Baghdad presso la corte del Califfo al-Ma' Mun, responsabile della Biblioteca Bayt al-Hikma (Casa della Sapienza). Autore del trattato Al-jabr w'al muqabala, da cui deriva il termine Algebra, di cui è considerato uno dei padri, insieme a Diofanto".

... se siete incuriositi, cercate in rete ulteriori dettagli.

Ma non abbiamo ancora detto "cosa significa"...

Algoritmo (informale)

- Specifica dei passi con cui si risolve un problema di natura arbitraria, utilizzando gli strumenti disponibili e le rispettive regole d'uso.

"Esempio classico (ascoltato e ri-ascoltato) delle ricette di cucina".

In questo corso gli algoritmi saranno "incorporati" in programmi per un calcolatore. Ma, in generale, il "calcolatore" potrebbe essere qualsiasi meccanismo

o animale, ed il "programma" una sequenza di azioni che esso è chiamato ad eseguire per raggiungere uno scopo determinato.

Ogni passo di un algoritmo richiede "tempo finito". Tuttavia, un algoritmo potrebbe nel suo complesso non terminare. Ciò accade quando uno o più passi vengono ripetuti un numero illimitato di volte.

Il seguente algoritmo "potrebbe" non terminare ... :-)

Alg 1.

1. Raggiungi piazza del Sapere.
2. Fai un giro a caso della piazza, sorridendo ai passanti.
3. Se nel corso del giro incontri l'anima gemella, fermati! Altrimenti, ricomincia dal passo 2.

Al contrario, l'algoritmo che segue termina sempre:

Alg. 2

1. Raggiungi l'entrata della mensa alle ore 12:00.
2. Sorridi ai passanti e, se incontri l'anima gemella, fermati e va ~~Via~~ con lei
3. Dalle 14:00, se non hai ancora incontrato l'anima gemella, rinuncia. Lascia la mensa e vai a casa.

Accetteremo solo gli algoritmi che terminano.

Ci preoccuperemo di valutare e di ridurre, per quanto possibile, la durata dell'esecuzione.

- Tra due algoritmi che risolvono lo stesso problema è da preferire quello che esegue "meno passi".

Algoritmica = studio degli algoritmi

Più formalmente

Input = dati in ingresso **Output** = dati in uscita

Problema computazionale: specifica della relazione tra Input e Output desiderata

Algoritmo: sequenza di passi computazionali che trasforma l'Input nell'Output

Istanza di un problema computazionale: particolare configurazione dell'Input.

Esempio: problema computazionale dell'ordinamento

Input: una sequenza di n numeri, $\langle a_1, a_2, \ldots, a_n \rangle$

Output: una permutazione (ri-arrangiamento) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ della sequenza $\langle a_1, a_2, \ldots a_n \rangle$ tale che $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Istanza del problema: $\langle 31, 41, 59, 16, 11, 5 \rangle$

(Input che soddisfa tutti i vincoli imposti dal problema, in questo caso "numeri".)

Un algoritmo si dice CORRETTO se, per ogni istanza di Input, l'algoritmo termina con Output corretto.

Un algoritmo corretto RISOLVE il problema computazionale dato.

Ambiti della vita quotidiana in cui sono utilizzati algoritmi: qualche esempio

- Comunicazioni nelle infrastrutture di reti
- BioInformatica
- Mondo economico-finanziario: transazioni, borse, aste...
- Protezione dei dati: confidenzialità, integrità
- Ricerca e recupero di informazioni tramite WEB.

⋮

... praticamente ovunque!

Preparazione a considerazioni sull'efficienza

Il dottor Fatto, il Dottor Volpe e ... la crescita esponenziale

Luogo d'incontro: Osteria del Gambero Rosso

Pinocchio ha un impiego modesto ma sicuro.
Risparmia uno zecchino d'oro e l'investe in
buoni del tesoro.

Incontra i consulenti finanziari, dottor Fatto e dott. Volpe,
che offrono un prodotto innovativo, "fatto apposta
per una persona intelligente come lei".

Il prodotto è un investimento presso la TRUE BANK
di St. Barbagian, ai Caraibi.

Pinocchio chiede se otterrà interessi più alti.

Il dottor Volpe: "con la nostra opzione l'istituto
aggiunge al suo versamento 10 zecchini all'anno!"

E ancora: "se anche i buoni fruttassero il 20%
all'anno il suo zecchino si moltiplicherebbe
solo per 1,2 ogni anno."

"Qualcuno può avergli detto che il ripetersi ogni anno della moltiplicazione implica una CRESCITA ESPONENZIALE, facendole balenare, con poca trasparenza, guadagni consistenti."

"Ma, mi creda, la nostra opzione è molto più remunerativa".

Pinocchio trascorre la notte a far conti...

		1° anno	2° anno	3° anno	4° anno	5° anno	10° anno
TRUE BANK	1	1+10	1+20	1+30	1+40	1+50	1+100
Buoni Tesoro	1	1,2	1,44	1,73	2,07	2,49	6,19

Profondamente convinto, al mattino Pinocchio corre a sottoscrivere...

Dopo pochi mesi Pinocchio prova a contattare la TRUE BANK per sapere come vanno le cose ma... da St. Barbagian nessuna risposta...

Ma Pinocchio ha fatto la scelta giusta?

$1 + K \cdot 10$
TRUE BANK

leggi di crescita

$1,20^K$
Buoni del Tesoro

K = numero di anni

(... Pinocchio non studia e non arriverebbe a questa formalizzazione.)

	$K=10$	$K=20$	$K=30$	$K=40$	$K=50$	...	$K=100$
$1+K\cdot 10$	101	201	301	401	501		1001
$(1,20)^K$	6,19	38,14	237,37	1469,77	9100,44		$\approx$ 83.000.000

Per un investimento più corto di 30 anni, ha fatto bene ...

ma per periodi più lunghi la scelta è pessima.

La funzione esponenziale quando supera un'altra funzione non esponenziale se ne stacca con una rapidità stratosferica!

Popolazione del pianeta

Supponiamo che ogni essere umano generi due esseri umani nel corso della vita (cioè, ogni famiglia, 4 figli)

Allora N genitori danno vita a $2N$ figli.

In generale, ci saranno

$N + 2N$ (genitori più figli)

$N + 2N + 2(2N) = N + 2N + 4N$ (gen., figli, nipoti)

$\vdots$

La K-esima generazione sarà costituita da $2^K \cdot N$ figli.

Ovviamente gli esseri umani, dopo un po', muoiono pure, ma le potenze di 2, cioè $2^0, 2^1, 2^2, 2^3 \ldots$ godono di un'importante proprietà: ognuna di esse supera di 1 la SOMMA di TUTTE le precedenti

$$2^4 = 2^3 + 2^2 + 2^1 + 2^0 \quad (+1)$$

2^{10} (più di 1000) $\qquad$ 2^{20} (più di un milione) $\qquad$ 2^{30} (più di un miliardo)

Pertanto, nell'espressione $2^K \cdot N$, l'elemento critico è 2^K (non N)

Basta aspettare che K cresca a sufficienza e questo elemento esplode.

Consiglio: quando incontrerete (o progetterete) un algoritmo che esegue un "numero di passi" esponenziale in K, dove K rappresenta la "lunghezza" dell'Input, prima di usarlo su Input di lunghezza K grande, ricordatevi di queste storielle e,... pensateci due volte!

Riferimenti:

- Le storielle raccontate (e diverse altre) si trovano in:

  "Algoritmi, divinità e gente comune"

  di F. Ruccio e R. Pagli.

  Edizioni ETS

- La formalizzazione dei concetti di problema computazionale, algoritmo ... etc, nel capitolo introduttivo di

  "Introduzione agli algoritmi e alle strutture dati", 3ª edizione, di

  T. Cormen, C. Leiserson, R. Rivest e C. Stein.

  Edizioni McGraw Hill

Problemi computazionali

"Ogni problema computazionale ben definito è risolvibile!"

"Il calcolatore può fare qualunque cosa: basta scrivere il programma adatto allo scopo!"

È vero? Ci crediamo?

Un po' di esempi per ragionare insieme ...

Numeri primi: 2, 3, 5, 7, 11, 13, 17...

"divisibili per se stessi e l'unità"

Problema computazionale: dato un intero $p > 1$

stabilire se è primo

Più formalmente:

Input: numero intero $p > 1$

Output: 1 se p è primo,
0 altrimenti

⟸

Problemi con output 0/1 sono detti problemi di DECISIONE

Algoritmo risolutore

```
PRIMO ( p)
    fattore = 2 ;
      while ( p % fattore != 0)        (fino a quando fattore non divide p)
          fattore = fattore + 1;
    RETURN (fattore == p)
```

Se p è primo è divisibile solo per se stesso e quindi PRIMO(p) restituirà 1.

Non è il "massimo" che si può fare ma ... risolve!

↓

Se p è un numero di K cifre, per semplicità supponiamo binarie, allora il valore di p è compreso tra $2^{K-1} \leq p \leq 2^K - 1$.

(Per esempio, un numero di 4 cifre è compreso tra 2^3 e $2^4 - 1$.)

Quindi, l'algoritmo PRIMO(p) controlla un numero esponenziale in K di fattori possibili.

Si potrebbe ottimizzare un po', per esempio evitando di controllare i fattori pari ... etc, ma si può far vedere che il numero di fattori che controlla resta esponenziale in K.

Tuttavia, per quanto ci riguarda:

- RISOLVE il problema
- termina sempre.

Congettura di Goldbach (XVIII secolo)

"Ogni numero intero $n \geq 4$ pari è la somma di due numeri primi p e q, cioè $n = p + q$."

Per esempio: $4 = 2+2$, $6 = 3+3$, $8 = 3+5$

$10 = 5+5$, $12 = 5+7$, $14 = 7+7$ ...

La congettura ad oggi non è stata né confutata con un controesempio né dimostrata (premio milionario per chi ci riuscisse)

Proviamo a risolvere la congettura per via algoritmica.

L'algoritmo che segue tenta di trovare un numero intero n pari per cui la proprietà NON è soddisfatta, cioè un n che non si può esprimere come somma di due primi.

Congettura Goldbach ( ) → UTILIZZA l'algoritmo PRIMO ( ) descritto in precedenza

```
n = 2;
DO {
    n = n + 2;
    CONTROESEMPIO = TRUE;
    FOR ( p = 2;  p ≤ n-2;  p = p+1)
    {
        q = n - p;
        IF ( PRIMO(p) && PRIMO(q))
            CONTROESEMPIO = FALSE;
    }
} WHILE ( ! CONTROESEMPIO)
RETURN n
```

Nota: se la congettura di Goldbach è vera, l'algoritmo NON termina. Se falsa, termina.

L'esempio ci mette in una condizione non tanto confortevole: vorremmo sapere, prima di mandarlo in esecuzione, se un programma termina o meno.

Problema computazionale (di decisione)

"Dato un programma arbitrario ed il suo input, è possibile stabilire se terminerà?"

Alan Turing ha dimostrato che non è risolvibile (indecidibile)

Non esiste alcun algoritmo che prende in input un programma generico ed il suo input e che dice se, una volta in esecuzione, terminerà o meno.

(1937, ispirato ai risultati di impossibilità di K. Gödel)

E come ha fatto Alan Turing a dimostrarlo?

Analizziamo le "idee di fondo" e la "prova".

In un calcolatore un programma è codificato mediante una sequenza di bit. Ogni programma, a sua volta, può essere visto come un INPUT per un altro programma (e.g., il sistema operativo che lo avvia e ne controlla l'esecuzione)

Pertanto, in generale, una programma è una sequenza di bit che può essere vista sia:

- come un programma da eseguire
- come un input per un altro programma

Supponiamo allora che il problema computazionale che stiamo considerando sia RISOLVIBILE.

⇓

$\exists$ un programma (algoritmo) TERMINA (A, D) che, presi in input un qualsiasi programma A ed il suo input D, restituisce in tempo finito un valore di verità T/F (0/1)

TRUE FALSE

Termina / Non Termina

A e D sono sequenze di bit.

Siamo noi a dare ad essi "significati" diversi.

Pertanto è legittimo invocare

TERMINA (A, A)

↳ usato come input al posto di D.

Costruiamo allora un altro programma che, al suo interno, invoca TERMINA $(\cdot,\cdot)$. Precisamente:

```
PARADOSSO ( A )

    WHILE ( TERMINA (A, A))
        ;
```

Si noti che:

- Se TERMINA $(A, A) = T$, allora PARADOSSO (A) non termina
- Viceversa, se TERMINA $(A, A) = F$, allora PARADOSSO (A) termina

Ma PARADOSSO $(\cdot)$ è un programma, cioè una sequenza di bit. Pertanto può essere esso stesso visto come un "input" ed è lecito invocare:

```
PARADOSSO ( PARADOSSO)

    WHILE ( TERMINA (PARADOSSO, PARADOSSO))
        ;
```

A questo punto:

- Se TERMINA (PARADOSSO, PARADOSSO) = T

↓

(cioè, PARADOSSO con input PARADOSSO termina)

allora PARADOSSO (PARADOSSO) non termina!

- Viceversa, se TERMINA (PARADOSSO, PARADOSSO) = F

↓

(cioè, PARADOSSO con input PARADOSSO non termina)

allora PARADOSSO (PARADOSSO) termina!

Conclusione: siamo di fronte ad un assurdo!

E poiché la deduzione logica che abbiamo eseguito è corretta, dobbiamo concludere che l'assunzione che esista TERMINA (A, D) è falsa!

TERMINA (A, D) non può esistere. Il problema è insolubile.

Il problema computazionale considerato è noto come "Problema della fermata" (Halting Problem)

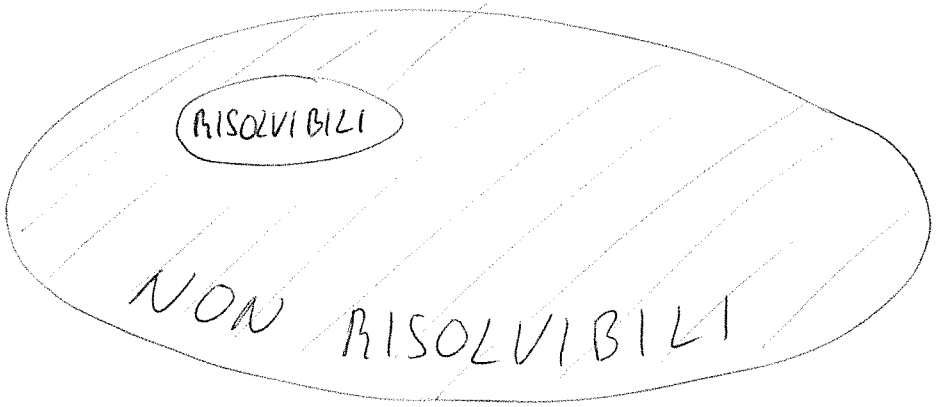
Osservazione aggiuntiva. Il risultato ottenuto è:

- INDIPENDENTE dal linguaggio di programmazione
- VALIDO per qualsiasi modello di calcolo che possa formalizzare il "comportamento" di un calcolatore.

Ritorneremo ampiamente su questi argomenti nel vostro corso di studi, in altri esami, con tempi e strumenti opportuni...

Per quanto ci riguarda, possiamo affermare con certezza che esistono problemi computazionali che NON sono risolvibili da un calcolatore.

In realtà è possibile dimostrare che i problemi computazionali risolvibili sono una "piccola parte" dell'Universo dei problemi computazionali



Riferimenti:

- Buona parte del materiale presentato oggi è stato estratto del capitolo introduttivo della precedente versione del libro di testo.

- Ai curiosi, consiglio una piacevole lettura sulla congettura di Goldbach (romanzo)

  "Zio Petros e la congettura di Goldbach"

  di A. Doxiadis (tascabili Bompiani)