

# Problemi, algoritmi e programmi

Antonella Santone

Anno Accademico 2008/2009

1

## Introduzione

• **Problema da risolvere**  
come trovare la stazione  
come cucinare l'anatra all'arancia



• **Procedimento per risolvere il problema**  
la strada più breve per la stazione  
una ricetta per l'anatra all'arancia



• **Agente di calcolo**  
il turista  
il cuoco



2

## Problem solving

Uno degli scopi fondamentali dell'informatica è:

**la risoluzione di problemi**

problema = compito che si vuole far risolvere automaticamente al **calcolatore**

agente di calcolo

3

## Problem solving (cont.)

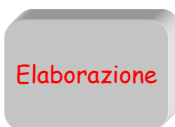
I problemi che siamo interessati a risolvere sono di natura molto varia:

- ❖ Trovare il maggiore tra due numeri
- ❖ Dato un elenco di nomi e numeri di telefono, trovare il numero di una data persona
- ❖ Dati  $a$  e  $b$ , risolvere l'equazione  $ax+b=0$
- ❖ Stabilire se una parola precede alfabeticamente un'altra
- ❖ Prenotare aerei, treni, hotel, ...
- ❖ Ordinare un elenco di nomi
- ❖ Ecc.

4

## Problem solving (cont.)

acquisire i dati

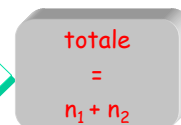
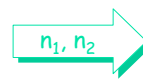


presentare i risultati



5

## Esempio: somma



6

## Attività per risolvere un problema

1. Comprendere il problema
2. Definire un procedimento risolutivo (*algoritmo*)
3. Implementare l'algoritmo in un linguaggio di programmazione
4. Prova
5. Documentazione
6. Manutenzione

7

## Comprendere il problema

- Focalizzare gli obiettivi
- Evidenziare  
le regole  
i dati espliciti ed impliciti
- Eliminare i dettagli inutili ed ambigui

8

## Attività per risolvere un problema

1. Comprendere il problema
2. Definire un procedimento risolutivo (*algoritmo*)
3. Implementare l'algoritmo in un linguaggio di programmazione
4. Prova
5. Documentazione
6. Manutenzione

9

## Algoritmo

descrizione rigorosa delle azioni da compiere per risolvere un problema di qualsiasi genere

### Esempi

la strada più breve per la stazione



una ricetta per l'anatra all'arancia



10

## Descrizione rigorosa

L'indicazione:

"prendi la II strada a destra e poi la I a sinistra"

può essere espressa in forma rigorosa come:

1. C'è una strada a destra? Se sì, vai al punto 3.
2. Vai avanti fino ad un incrocio; vai al punto 1.
3. Vai avanti fino ad un incrocio.
4. C'è una strada a destra? Se no, vai al punto 3.
5. Svolta a destra.
6. Vai avanti fino ad un incrocio.
7. C'è una strada a sinistra? Se no, vai al punto 6.
8. Svolta a sinistra.

11

## Considerazioni

1. Suddivisione in sottoproblemi
2. Precisione dell'algoritmo cresce al diminuire delle capacità dell'agente di calcolo

12

## Caratteristiche di un algoritmo

- Lunghezza finita
- Struttura in passi elementari  
(direttamente eseguibile dall'agente di calcolo)
- Non casualità  
(risultato certo e ripetibile)
- Determinismo
- Non ambiguità
- Comprensibilità

Esempio di ambiguità:

bagnare i capelli, applicare lo shampoo, sciacquare e ripetere una seconda volta

Ripetere cosa??

13

## Struttura in passi elementari

Tutte le operazioni specificate dall'algoritmo devono essere eseguibili dall'agente (operazioni elementari) ... altrimenti è necessario scomporre il problema troppo complesso in sotto problemi più semplici

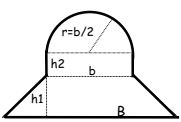
Rompere il guscio di un uovo??



Colpire con un gesto secco ma leggero il guscio dell'uovo con il dorso di un coltello. Tenendo verticale l'uovo, aprire il guscio inserendo l'unghia del pollice nell'incavatura formatasi nel guscio

14

## Esempio: l'area di una campana



Area della campana =

$$A1 + A2 + A3$$



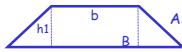
Sottoproblema 1

$$A1 = \frac{1}{4} \pi r^2$$



Sottoproblema 2

$$A2 = b h2$$



Sottoproblema 3

$$A3 = b h1 +$$

$$\frac{1}{2} \left( \frac{1}{2} (B-b) h1 \right) +$$

$$\frac{1}{2} \left( \frac{1}{2} (B-b) h1 \right)$$

15

## Esempio: gestione biblioteca



- ❖ Libri disposti sugli scaffali
- ❖ La posizione di ogni libro è fissa ed è individuata da due coordinate:
  - numero dello scaffale
  - posizione nello scaffale
- ❖ La biblioteca è dotata di uno schedario (ordinato per autore/i e titolo). Ogni scheda contiene, nell'ordine:
  - Cognome e nome dell'autore
  - Titolo del libro
  - Numero dello scaffale
  - Posizione attribuita al libro nello scaffale

16

## Esempio di scheda

Autore: *Manzoni*

Titolo: *I promessi Sposi*

Scaffale: 33

Posizione: 13

17

## Problema

Trovare un libro??



18

## Formulazione dell'algoritmo

1. Cerca la scheda del libro nello schedario
2. Segnati numero scaffale e posizione
3. Cerca lo scaffale indicato
4. Accedi alla posizione indicata e preleva il libro

Questo modo incrementale di procedere si dice **top-down** o anche procedimento per **raffinamenti successivi**

## Primo sotto-algoritmo di ricerca

1. Prendi la prima scheda dello schedario
2. Se titolo e autore/i sono quelli cercati, la ricerca termina con successo, altrimenti passa alla scheda successiva
3. Continua di scheda in scheda finché non trovi quella cercata. Se vengono esaurite le schede, il libro cercato non esiste. Devi cercare il libro in un'altra biblioteca.

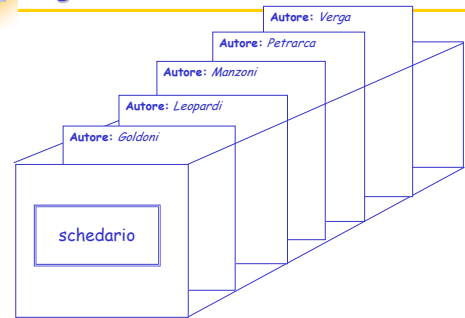
20

## Cosa succede se l'autore cercato è Verga?



21

## Cosa succede se l'autore cercato è Verga?



22

## Secondo sotto-algoritmo di ricerca

1. Esamina la scheda centrale dello schedario
2. Se la scheda centrale corrisponde al libro cercato allora la ricerca termina
3. In caso contrario, prosegui allo stesso modo nella metà superiore o inferiore dello schedario a seconda che il libro cercato segua o preceda quello indicato sulla scheda

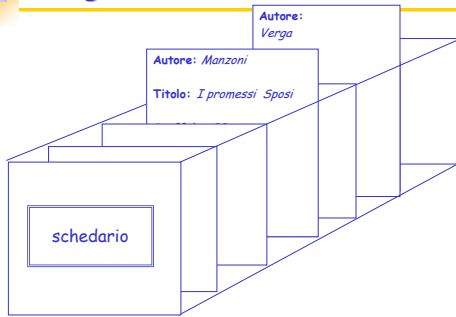
23

## Cosa succede ora se l'autore cercato è Verga?



24

## Cosa succede ora se l'autore cercato è Verga?



25

## Secondo sotto-algoritmo di ricerca: errore

1. Esamina la scheda centrale dello schedario
2. Se la scheda centrale corrisponde al libro cercato allora la ricerca termina
3. In caso contrario, prosegui allo stesso modo nella metà superiore o inferiore dello schedario a seconda che il libro cercato segua o preceda quello indicato sulla scheda

L'algoritmo è incompleto:

c'è un'altra condizione di terminazione quando il libro non esiste

26

## Revisione del passo 2

Terminazione

Se la scheda centrale corrisponde al libro cercato

oppure

Se la parte dello schedario da consultare è vuota

Libro trovato



Libro inesistente



27

## Qualità degli algoritmi

Due qualità fondamentali di un algoritmo:

- **Correttezza**  
l'algoritmo permette effettivamente di risolvere il problema
- **Efficienza**  
l'esecuzione dell'algoritmo richiede un uso limitato di risorse  
un algoritmo è tanto più efficiente quanto meno risorse richiede per la sua esecuzione. Una risorsa importante è il **tempo di esecuzione**

28

## Esempio: gestione biblioteca

- Entrambi gli algoritmi di ricerca sono corretti
- Il secondo algoritmo è più efficiente del primo

29

## Altre qualità degli algoritmi

- **leggibilità**  
essere facilmente comprensibile
- **modificabilità**  
essere facilmente modificabile, a fronte di (piccole) modifiche nelle specifiche del problema risolto dall'algoritmo
- **parametricità**
- **riusabilità**

30

## Esempio: algoritmo del risveglio

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. Prendere il bus per andare a scuola

### NOTA

I passi sono eseguiti in sequenza e l'ordine delle istruzioni è essenziale per la correttezza dell'algoritmo



31

## ... riassumendo

### Algoritmo

Sequenza di azioni che trasforma i dati iniziali in un numero finito di passi, elementari e non ambigui, per giungere al risultato finale

Questa sequenza di azioni può essere eseguita da un agente di calcolo

32

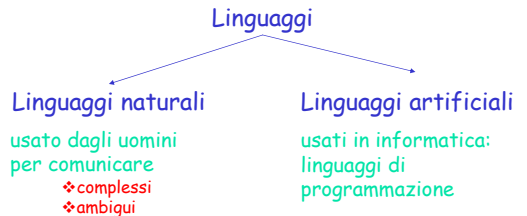
## Attività per risolvere un problema

1. Comprendere il problema
2. Definire un procedimento risolutivo (*algoritmo*)
3. Implementare l'algoritmo in un linguaggio di programmazione
4. Prova
5. Documentazione
6. Manutenzione

33

## Implementazione di un algoritmo

Algoritmo deve essere trascritto in un linguaggio



34

## Alcuni linguaggi di programmazione "famosi"

- FORTRAN: metà degli anni '50 (FORMula TRANslator)
- COBOL: metà degli anni '50 (Common Business-Oriented Language)
- Pascal: inizio degli anni '70 (nome dal matematico francese Blaise Pascal che fu il primo ad ideare una macchina calcolatrice: la Pascalina)
- C: inizio degli anni '70  
suo antenato: linguaggio B
- Prolog: inizio degli anni '70 (PROgramming in LOGic)
- ...

Linguaggi ad alto livello

35

## Il linguaggio C

- ❖ Il linguaggio C è stato sviluppato intorno al 1972, nei Bell Laboratories AT&T americani, da Dennis Ritchie
- ❖ E' nato come linguaggio di sviluppo del Sistema Operativo UNIX

36

## Sintassi e semantica

### Linguaggio

#### Sintassi

insieme di regole che consentono di costruire correttamente le frasi del linguaggio

#### Semantica

disciplina che studia il significato delle parole e delle frasi

LIVELLO	FORMA CORRETTA	FORMA NON CORRETTA
sintattico	sono andato a scuola	ho andato a scuola
semantico	il gatto è un animale	l'albero è un animale

37

## Programma

Testo (i.e. sequenza di istruzioni) scritto in accordo alla *sintassi* e *semantica* di un linguaggio di programmazione

38

## Linguaggio macchina

Un calcolatore non è in grado di eseguire direttamente programmi scritti in linguaggi ad alto livello

un calcolatore è in grado di eseguire direttamente solo programmi scritti nel proprio linguaggio macchina



```
0110001000100101
1101001001010101
0010100101000100
1110100100110101
.....
```

39

## Linguaggio macchina (cont.)

Il linguaggio macchina è

- ❖ linguaggio di programmazione comprensibile direttamente dal calcolatore
- ❖ molto elementare e primitivo: sequenza di cifre binarie.  
Ad esempio, una possibile istruzione potrebbe essere 0010110000101100
- ❖ è difficile da comprendere per un essere umano
- ❖ specifico di un calcolatore: calcolatori di tipo diverso hanno linguaggi macchina differenti

40

## Traduzione

Per rendere un programma (scritto in un linguaggio ad alto livello) eseguibile da un calcolatore è necessario tradurre il programma in un programma **equivalente** scritto nel linguaggio macchina del calcolatore

La traduzione può avvenire in due modi:

- ❖ compilazione
- ❖ interpretazione

41

## Compilazione

un programma scritto in un linguaggio di programmazione ad alto livello viene trasformato in un programma eseguibile e poi può essere eseguito più volte senza doverlo ricompilare nuovamente il programma



Programma scritto in C

Compilazione



```
0110001000100101
1101001001010101
0010100101000100
1110100100110101
```

codice eseguibile

### ERRORI

- sintassi (compilatore)
- d'esecuzione (divisione per 0)
- logici

controllo della correttezza sintattica

Nota: correttezza sintattica

correttezza semantica

42

## Interpretazione

Traduzione istruzione per istruzione:

ciascuna istruzione del programma scritto in un linguaggio di programmazione ad alto livello viene trasformata in istruzioni del linguaggio macchina ed eseguita

43

## Compilazione & interpretazione

Una analogia con la traduzione tra linguaggi diversi

- ❖ la compilazione è analoga alla traduzione di un libro
- ❖ l'interpretazione è analoga alla traduzione simultanea

44

## Compilazione & interpretazione

- ❖ Linguaggi compilati  
Pascal, C
- ❖ Linguaggi interpretati  
Prolog, Lisp

45

## Vantaggi & svantaggi: compilazione

Vantaggi

- ❖ creazione di programmi eseguibili
- ❖ velocità d'esecuzione del programma

Svantaggi

- ❖ correzioni solitamente al termine della compilazione
- ❖ impossibilità di controllare il funzionamento solo di una parte del programma

46

## Vantaggi & svantaggi: interpretazione

Vantaggi

- ❖ immediata visione dell'errore (l'interprete controlla immediatamente la sintassi)
- ❖ possono essere eseguite parti di programmi anche non completi

Svantaggi

- ❖ lenta esecuzione
- ❖ il programma eseguibile dipende dall'interprete perché il codice tradotto non si può memorizzare

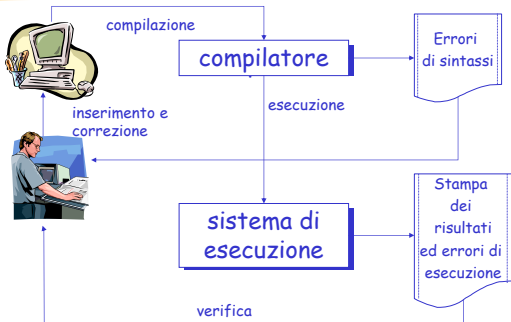
47

## Attività per risolvere un problema

1. Comprendere il problema
2. Definire un procedimento risolutivo (*algoritmo*)
3. Implementare l'algoritmo in un linguaggio di programmazione
4. Prova
5. Documentazione
6. Manutenzione

48

## Prova



49

## Attività per risolvere un problema

1. Comprendere il problema
2. Definire un procedimento risolutivo (*algoritmo*)
3. Implementare l'algoritmo in un linguaggio di programmazione
4. Prova
5. Documentazione
6. Manutenzione

50

## Documentazione

- Scrivere un manuale d'uso che accompagni il programma
- Questo manuale deve essere scritto facendo uso della terminologia tipica del problema e non in gergo computeristico

51

## Attività per risolvere un problema

1. Comprendere il problema
2. Definire un procedimento risolutivo (*algoritmo*)
3. Implementare l'algoritmo in un linguaggio di programmazione
4. Prova
5. Documentazione
6. Manutenzione

52

## Manutenzione

Non esistono parti del programma soggette a usura!!

Manutenzione del programma =  
modificarlo  
aggiornarlo  
estenderlo  
renderlo più veloce  
...

*Per gli studenti il problema della manutenzione è relativamente ridotto: una volta che il programma funziona si passa all'esame successivo*



53

## Riepilogo

Problemi → Algoritmi → Programmi

**Problema:** compito che si vuole far risolvere automaticamente al calcolatore

**Algoritmo:** descrizione rigorosa delle azioni da compiere per risolvere un problema

**Programma:** sequenza di istruzioni in un linguaggio di programmazione. **Scopo del programma:** fornire al calcolatore le capacità per risolvere un dato problema



54