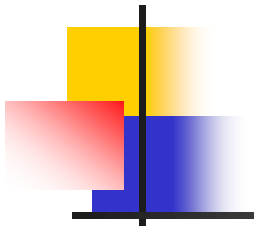


Dal problema al programma: introduzione al problem-solving in linguaggio C

Capitolo 1: Introduzione

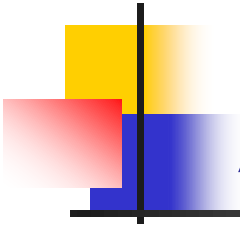
G. Cabodi, P. Camurati, P. Pasini, D. Patti, D. Vendraminetto

MAGGIOLI EDITORE APOGEO education 2016

Il problem-solving

Attività del pensiero che:

- un organismo o un dispositivo di intelligenza artificiale
- mettono in atto
- per raggiungere una condizione desiderata a partire da una condizione data
- altamente creativa, di natura progettuale.



Approccio al problem-solving

1. analisi del problema:

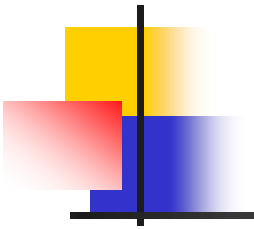
   lettura delle specifiche, comprensione del problema, identificazione della classe di problemi noti cui il problema in esame appartiene

2. identificazione delle metodologie:

   scelta tra paradigmi algoritmici noti (incrementale, divide et impera, bottom-up, greedy, etc.)

3. selezione di un approccio:

   scelta dell'approccio migliore in base ad un'analisi di complessità



4. decomposizione in sottoproblemi:

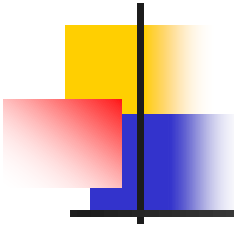
   identificazione dei sottoproblemi e delle loro interazioni in un’ottica di modularità

5. definizione dell'algoritmo risolutivo:

   identificazione della sequenza di passi elementari e dei dati su cui opera e dimostrazione della correttezza

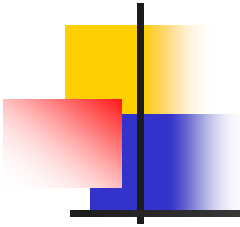
6. riflessione critica:

   ripensamento per identificare criticità, possibili migliorie, etc.



Problemi computazionali

- modelli formali cui è associato un insieme di domande a cui dare risposta mediante le elaborazioni di un computer che esegue un programma
- insieme di istanze infinite del problema, a ciascuna delle quali è associata una soluzione.
- istanza: problema su dati specifici.



Esempio

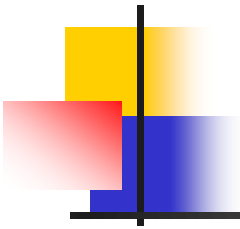
- Formulazione del problema:

  calcolare la somma di tutti gli interi non negativi minori di un numero naturale dato N

- Soluzione:

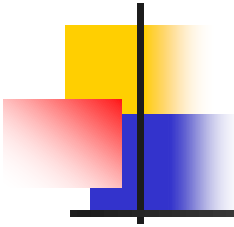
  { <0, 0> , <1, 1>, <2, 3>, <3, 6>, <4, 10>, ... }

  dove la coppia <x, y> significa che la somma dei numeri interi non negativi minori o uguali a x è y.



Criteri di classificazione

- Approccio usuale: in base alle strutture dati e/o ai costrutti di controllo utilizzati
- Approccio alternativo più vicino all'utente e più lontano dal programmatore in base:
  - alla tipologia del problema
  - all'ambito applicativo: tipo di informazioni trattate e classe di problema
  - alla struttura dati utilizzata
  - alle strategie algoritmiche.



Tipologie di problemi

- decisione:

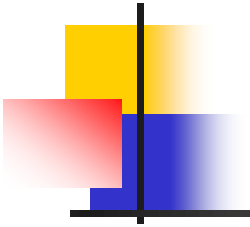
  problemi per cui la risposta a ogni istanza è binaria.

  Esempio: dato un $n > 1$, n è primo?

- ricerca:

  la risposta a ogni istanza è una stringa di bit, che rappresenta un'opportuna informazione.

  Esempio: ordinare un insieme di dati distinti su cui è definita una relazione d'ordine



- verifica:

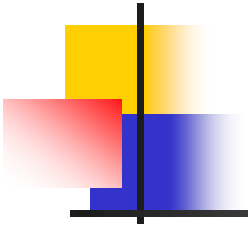
  data un'istanza e una soluzione presunta (certificato), si appura se il certificato è davvero una soluzione per l'istanza.

  Esempio: data una funzione booleana *f* e un'assegnazione di valori alle variabili, si verifica che *f* valga 1 per tale assegnazione

- selezione:

  caso particolare di verifica: date soluzioni e criterio di accettazione, separano le soluzioni che soddisfano/non soddisfano il criterio

  Esempio: dati i risultati di studenti, identificare tutti e soli gli studenti il cui risultato è sopra la media



- simulazione:

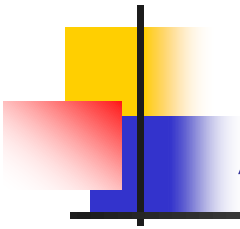
  rappresentazione interattiva della realtà basata sulla costruzione di un modello computazionale di un sistema del quale si vuole analizzare il funzionamento.

  Esempio: dati n sportelli e un'ipotesi di arrivo di clienti, ciascuno con un servizio di durata temporale nota, prevedere i tempi di attesa

- ottimizzazione:

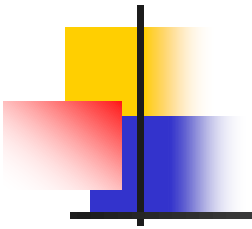
  data un'istanza e una funzione di costo o vantaggio, tra più soluzioni possibili si seleziona quella a costo minimo o a vantaggio massimo.

  Esempio: dato contenitore con capacità, dati oggetti con volume e valore, trovare l'insieme di oggetti compatibili con la capacità di valore complessivo massimo .



Altri criteri di classificazione

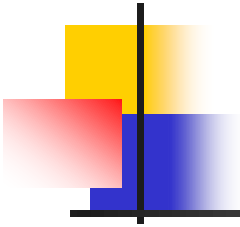
- Ambito applicativo: problemi
  - numerici
  - matematici
  - elaborazione di testi
  - elaborazione di informazioni non numeriche
  - etc.
- Natura dei dati primitivi:
  - scalari e/o aggregati
  - vettoriali



- Tipi di dato:
  - numerici (interi o reali) o
  - carattere (caratteri o stringhe)
  - tipi di dato astratto
- Costrutti del linguaggio usati:
  - elementari (costrutti condizionali, costrutti iterativi semplici o annidati)
  - avanzati e/o funzioni.

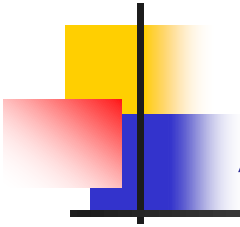
Classificazione adottata

	Senza vettori	**Con vettori/matrici**
Numerici	Equaz. 2° grado Serie e successioni numeriche …	Statistiche per gruppi Operazioni su insiemi di numeri Generazione numeri primi Somme/prodotti matriciali
Codifica	Conversioni di base (es. binario/decimale) Crittografia di testo …	Conversioni tra basi numeriche Ricodifica testi utilizzando tabelle di conversione
Elab. Testi	Manipolazione stringhe Menu con scelta Grafico di funzione (asse X verticale)	Conteggio caratteri in testo Grafico funzione (asse X orizzontale) Formattazione testo (centrare, eliminare spazi)
Verifica/ selezione	Verifica di ordinamento/congruenza di dati Verifica mosse di un gioco Filtro su elenco di dati Ricerca massimo o minimo Ordinamento parziale	Verifica di unicità (o ripetizione) di dati Selezione di dati in base a criterio di accettazione Ricerca di dato in tabella (in base a nome/stringa) Ordinamento per selezione



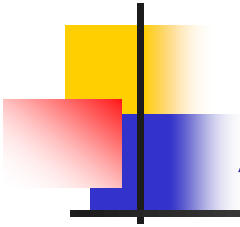
Passi per la soluzione

- identificazione non ambigua e completa del problema da risolvere con analisi ed eventuale completamento delle specifiche
- costruzione di un modello formale del problema
- definizione di un algoritmo di risoluzione e analisi di complessità
- codifica dell'algoritmo in un linguaggio di programmazione
- validazione dell'algoritmo e della sua implementazione su istanze significative.



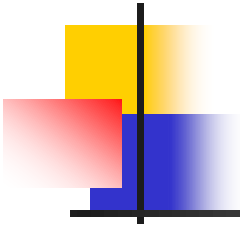
Algoritmo

- Un algoritmo (da Al-Huarizmi, matematico di area persiana del IX secolo d.C.) è una sequenza finita di istruzioni che:
  - risolvono un problema
  - soddisfano i seguenti criteri
    - ricevono valori in ingresso
    - producono valori in uscita
    - sono chiare, non ambigue ed eseguibili
    - terminano dopo un numero finito di passi
  - operano su strutture dati.



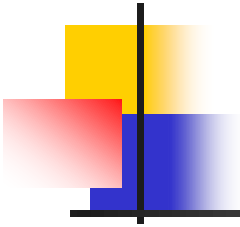
Algoritmo = strategia

- Scegliere un algoritmo spesso significa individuare la miglior strategia per risolvere un problema
- La scelta si basa su:
  - conoscenza delle operazioni elementari e delle funzioni di libreria fornite dal linguaggio C
  - conoscenza dei costrutti linguistici (tipi di dato, costrutti condizionali e iterativi)
  - esperienza su tipi diversi di problemi.



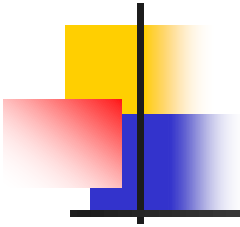
Strategia

- La maggioranza dei problemi risolti mediante programmi consiste nell'elaborazione di informazioni ricevute in input, per produrre risultati in output
- La parte più rilevante è l'elaborazione, che richiede:
  - l'individuazione di dati (risultati intermedi)
    - che possono essere scalari e/o aggregati
  - la formalizzazione di passi (operazioni) necessarie a valutare i risultati intermedi (a partire da altri dati):
    - i passi intermedi sono spesso formalizzati in termini di costrutti condizionali e/o iterativi. Possono poi essere modularizzati mediante funzioni.



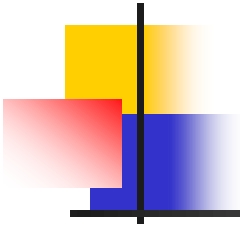
In pratica!

- L’esperienza è un requisito fondamentale (come in molte altre discipline) per una efficace scelta di strategie risolutive: un algoritmo (un programma) è in definitiva un progetto
- Lo studio di problemi classici già risolti è un buon passo di partenza
- Talvolta, in mancanza di altri strumenti, un valido punto di partenza è l’analisi della soluzione “a mano” o “su carta” del problema, facendo corrispondere carta (o lavagna) a variabili e calcoli ed espressioni.



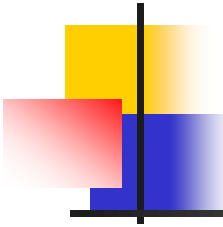
Struttura dati

- La scelta della struttura dati deve tener conto:
  - della natura del problema, delle informazioni ricevute in input, dei risultati richiesti
  - delle scelte algoritmiche
- Scelta di una struttura dati = decidere quali (di che tipo) e quante variabili saranno necessarie per immagazzinare dati in input, intermedi e risultati
- Talvolta la struttura dati può essere scelta prima di definire l'algoritmo, ma spesso è necessario considerare dati e algoritmo insieme.



Scelta della struttura dati

- Consiste in:
  - individuare i tipi di informazioni da rappresentare (input, dati intermedi, risultati): numeri (interi o reali), caratteri o stringhe, `struct`
  - decidere se è necessario collezionare i dati in vettori o matrici (aggregati numerabili) oppure se sono sufficienti dati scalari (o `struct`)
- Alcuni problemi si risolvono con poche istruzioni (con costrutti condizionali) su dati scalari
- Molti problemi richiedono iterazioni su dati.



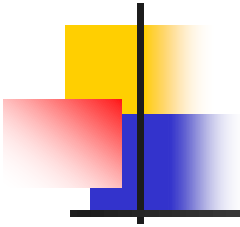
Problemi iterativi e vettori

- Un problema iterativo non richiede un vettore quando è sufficiente manipolare il generico (l'i-esimo) dato, senza **"ricordare"** i precedenti

  Esempio: sommatoria, ricerca del massimo

- Un problema iterativo richiede un vettore se è necessario raccogliere/collezionare (in variabili) tutti i dati, prima di poterli manipolare

  Esempio: input di dati, output in ordine inverso



Scelta dell'algoritmo

- Individuare l'algoritmo (costrutti condizionali e/o iterativi) può essere decisamente semplice (suggerito direttamente dal problema)

  Esempio : problemi numerici/matematici

- In altri casi scegliere un algoritmo consiste nel realizzare un vero progetto, confrontando strategie diverse, valutando pro e contro (vantaggi e costi)

  Esempio: pianificazione di attività in funzione di criteri di ottimalità.