Gli algoritmi

La soluzione algoritmica di problemi

- Algoritmo, il concetto fondamentale e centrale dell'informatica
  - da AL-KHOWARIZMI (825 dc). Una procedura per risolvere un problema matematico in un numero finito di passi che implicano frequenti ripetizioni di un'operazione.
  - In senso lato, una procedura che eseguita a passo a passo risolve un problema.
    - Dal dizionario Webster

Esempi

- Algoritmi (o procedure):
  - per calcolare il Massimo Comun Divisore
  - per costruire modellini di aerei (espressi nei fogli di istruzioni)
  - per azionare la lavatrice
  - per suonare una melodia al piano (espressa in un insieme di simboli negli spartiti)

Algoritmo di Euclide

- Dati due numeri interi e positivi *m* e *n*, calcola il più grande intero che li divide

1. dividere *m* per *n*, e sia *r* il resto della divisione (con $0 \le r \le n$)
2. se r = 0 allora la risposta è *n* e STOP
3. porre $m \leftarrow n$ e $n \leftarrow r$ e ripetere il passo 1.

Verifica empirica

- **Funziona davvero? Il ciclo dei 3 passi termina sempre?**
- **Proviamo con $m \leftarrow 119$ e $n \leftarrow 54$:**

1. 119 / 544 dà resto $r = 119$
2. se $r \neq 0$ allora
3. poniamo $m \leftarrow 544$ e $n \leftarrow 119$. Torniamo al passo 1.
1. 119 / 544 dà resto $r \leftarrow 119$
2. se $r \neq 0$ allora
3. poniamo m $\leftarrow$ 119 e n $\leftarrow$ 68 Torniamo al passo 1.
1. 119 / 68 dà resto $r \leftarrow 51$
2. se $r \neq 0$ allora
3. poniamo $m \leftarrow 68$ e $n \leftarrow 51$ Torniamo al passo 1.
1. 68 / 51 dà resto $r \leftarrow 17$
2. se $r \neq 0$ allora
3. poniamo m $\leftarrow$ 51 e n $\leftarrow$ 17 Torniamo al passo 1.
1. 51 / 17 dà resto $r \leftarrow 0$
2. se $r = 0$ allora il MCD = $n = 17$

Algoritmo vs procedura di calcolo

- Parliamo di algoritmi solo se definiscono un insieme finito di istruzioni che danno luogo a una sequenza finita di operazioni
- Il processo di ripetizione di cicli deve terminare
  - Nell'algoritmo di Euclide, la sequenza dei resti *r*
    - sequenza di numeri interi positivi decrescenti
- Non tutte le procedure soddisfano questo requisito
  - Si parla allora di procedure di calcolo

Trovare algoritmi

- La ricerca di algoritmi è stata una grande parte del lavoro dei matematici nei secoli
- Gli algoritmi rendono il lavoro più semplice
- La loro esecuzione non richiede la comprensione dei principi su cui si fonda
- Il computer è un esecutore di algoritmi

I linguaggi di programmazione

- L'algoritmo va rappresentato in un sistema formale per essere comunicato alla macchina che lo esegue
- I linguaggi di programmazione
- Una varietà di approcci per affrontare il processo della programmazione

Sintassi

- Sistema di regole formali che definiscono il linguaggio
- Consente di stabilire se un'istruzione è ben formata
  - È corretto scrivere: **se *x* + *n* > *m* allora STOP** ?
- Facilitano il compito al programmatore
- I programmi devono essere tradotti nel linguaggio nativo della macchina

Diversi livelli di espressività

- In un linguaggio ad alto livello, PASCAL
  - **semiperimetro := base + altezza;**
- In un linguaggio assembly, simbolico del linguaggio macchina,
  - **LOAD x**
  - **ADD y**
  - **STORE z**

Linguaggi di basso livello

- In un linguaggio macchina, ogni istruzione è una sequenza di cifre binarie
  - Totalmente illeggibile per l'uomo
  - Perfettamente non ambiguo per la macchina

```
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
```

N.B. traduzione delle istruzioni precedenti

Traduzione dei linguaggi

- Il concetto di traduzione dei linguaggi ha permesso l'evoluzione dei linguaggi verso sistemi simbolici più espressivi e più facilmente manipolabili dai programmatori
- Il programmatore scrive un programma in un linguaggio ad alto livello senza preoccuparsi della macchina che esegue il programma
- Il compilatore viene predisposto per una macchina specifica e poi traduce tutti i programmi scritti in uno specifico linguaggio ad alto livello

Programma compilatore

- Il compilatore traduce istruzioni scritte in un linguaggio ad alto livello, definite da regole sintattiche precise
- Necessità di strumenti formali per la descrizione delle grammatiche (tavole sintattiche, grammatiche, BNF)
- Programma sorgente: il programma ad alto livello tradotto dal compilatore
- Programma oggetto: il risultato della traduzione del compilatore

Influenza del modello sui linguaggi

- Concetto di istruzione
  - l'unità di base del programma, memorizzate in successive celle di memoria
- Concetto di sequenzialità e iterazione
  - In diversi costrutti dei linguaggi e in tutto il processo di esecuzione
- Concetto di variabile
  - Le celle di memoria contenenti i valori da manipolare

È sorprendente che il computer di Von Neumann sia rimasto così a lungo il paradigma fondamentale dell'architettura dei calcolatori.

Ma dato il fatto, non è sorprendente che i linguaggi imperativi siano I principali oggetti di studio e sviluppo.

Perchè come Backus ha sottolineato i linguaggi imperativi hanno solide radici nell'architettura della macchina di Von Neumann e ne sono l'immagine.

Horowitz (Fundamentals of Programming Languages, 1983)

Evoluzione dei linguaggi

- Anni ’50: i primi linguaggi ad alto livello
  - **FORTRAN**, introduce il concetto di sottoprogrammi che operano su dati comuni
  - **ALGOL**, introduce il concetto di struttura dei programmi e di procedure ricorsive, ovvero che richiamano se stesse
  - **COBOL**, introduce il concetto di FILE e di descrizione dei dati

Evoluzione dei linguaggi

- Anni ’60: i linguaggi orientati al problema
  - **LISP**, introduce uniformità tra dati e programmi e un paradigma di programmazione basato sul concetto di funzione
  - **APL**, linguaggio matematico, ricco di notazioni e operatori per operare su strutture come vettori e matrici
  - **SNOBOL**, offre strumenti utili per la manipolazione di sequenze di caratteri

Evoluzione dei linguaggi

- Anni ’70: metodologia di programmazione
  - **PASCAL**, ha lo scopo di insegnare la programmazione strutturata, una possbile risposta alla necessità di programmare con un metodo. Fa seguito **Modula-2** che introduce il concetto di modulo.
  - **C**, linguaggio ad alto livello con visibilità e accesso alla macchina
  - **Prolog**, linguaggio basato sulla logica, non convenzionale diventato di nicchia

Evoluzione dei linguaggi

- Anni ’80: programmazione in grande, esigenza di modularità e di astrazione
  - La programmazione a oggetti. Una classe definisce un insieme di oggetti e le procedure per manipolarli
  - SmallTalk, Objective-C, C++

Teoria degli algoritmi

- Gli algoritmi risolvono un problema
  - Possibilmente sono corretti ...
  - Dimostrazioni di correttezza degli algoritmi vs verifica empirica
  - Complessità degli algoritmi
    - Se gli algoritmi sono corretti, sono *buoni*?
    - Criteri di bontà: efficienza nell’uso delle risorse sia di tempo di calcolo, sia di occupazione di memoria
  - Problemi intrinsecamente difficili