Corso di Fondamenti di Informatica

Corso di Laurea in Ingegneria Elettronica

Prof. Diego Calvanese

Anno Accedemico 2001/2002

Introduzione all'elaborazione automatica delle informazioni

Algoritmi e programmi

Problemi da risolvere usando elaboratori possono essere di natura molto varia.

Esempi

1. Dati due numeri, trovare il maggiore.
2. Dato un elenco di nomi e numeri di telefono (rubrica o elenco telefonico) e un nome, trovare il numero di telefono corrispondente.
3. Data la struttura di una rete stradale e le informazioni sui flussi dei veicoli, determinare il percorso più veloce da A a B.
4. Scrivere tutti i numeri pari che non sono la somma di due numeri primi (Congettura di Goldbach).
5. Decidere per ogni programma C e per ogni dato in ingresso, se il programma C termina quando viene eseguito su quel dato.

Caratteristica comune ai problemi

informazioni in ingresso $\Longrightarrow$ informazioni in uscita

Osservazioni sulla formulazione dei problemi:

- descrizione non fornisce un metodo risolutivo (si pensi all'esempio 3)
- descrizione del problema è talvolta **ambigua** o **imprecisa** (ad esempio, 2 con Mario Rossi che compare più volte)
- per alcuni problemi **non è noto un metodo risolutivo** (ad esempio 4)
- esistono problemi per i quali è stato dimostrato che **non può esistere un metodo risolutivo** (ad esempio 5)

Noi consideriamo solo problemi per i quali è noto esistere un metodo risolutivo.

Per delegare ad un calcolatore la soluzione di un problema è necessario:

1. Individuare un **algoritmo** che risolve il problema, ovvero un insieme di passi che, eseguiti in ordine, permettono di calcolare i risultati a partire dalle informazioni a disposizione.

   Proprietà di un algoritmo:

   non-ambiguità: le istruzioni devono essere univocamente interpretabili dall'esecutore

   eseguibilità: ogni istruzione deve poter essere eseguita (in tempo finito) con le risorse a disposizione

   finitezza: l'esecuzione dell'algoritmo deve terminare in tempo finito per ogni insieme di dati in ingresso

2. **Rappresentare in un linguaggio di programmazione (LDP)**

2. l'algoritmo		2. programma
2. le informazioni a disposizione		2. dati in ingresso
2. le informazioni utilizzate dall'algoritmo	$\Longrightarrow$	2. dati ausiliari
2. le informazioni fornite al termine		2. dati in uscita

Riassumendo:

problema $\Longrightarrow$ **algoritmo** $\Longrightarrow$ **programma**

Individuare un algoritmo per un dato problema può essere molto complesso:
$\Longrightarrow$ Conviene operare per **livelli di astrazione**:

- si parte da una versione molto generale
- si **raffinano** via via le varie parti dell'algoritmo fino ad ottenere una descrizione dettagliata che può essere direttamente codificata in un LDP

$\Longrightarrow$ **metodo dei raffinamenti successivi**

Pseudocodice per la specifica di algoritmi

- si usano **frasi in linguaggio naturale** che esprimono operazioni o condizioni
- operazioni vengono eseguite in sequenza, tranne che per le . . .
- **strutture di controllo** dell'esecuzione delle operazioni (specificate usando parole chiave)

if condizione	**for** ogni valore compreso tra . . . e . . .
then operazione 1	**do** operazione
else operazione 2	
while condizione	**do** operazione
do operazione	**while** condizione

Lo pseudocodice si presta bene al metodo dei raffinamenti successivi: un **raffinamento** consiste nel sostituire un'operazione con

- una sequenza di operazioni, oppure
- una struttura di controllo

Esempio: Calcolo del massimo comun divisore di due interi positivi m ed n.

Algoritmo

1. calcola l'insieme I dei divisori di m
2. calcola l'insieme J dei divisori di n
3. calcola l'insieme K intersezione di I e J (divisori comuni)
4. restituisci il valore massimo tra quelli in K

Raffinamento del passo 1

1.1 inizializza I all'insieme vuoto
1.2 **for** ogni numero i compreso tra 2 ed m
do if i è divisore di m
then aggiungi i ad I

Esercizio: scrivere un algoritmo per il calcolo del MCD che si basi sulla seguente proprietà

$$MCD(m,n) = \begin{cases} m, & \text{se } m = n \\ MCD(m-n,n), & \text{se } m > n \\ MCD(m,n-m), & \text{se } m < n \end{cases}$$

Programmi: rappresentano algoritmi in un linguaggio di programmazione

Esempio: Programma che

1. legge due numeri da tastiera
2. ne calcola il MCD e
3. lo stampa

```
/* File: mcd2.c */
#include <stdio.h>
int main(void)
{ int m, n;
  printf("Immetti due interi positivi: ");
  scanf("%d%d", &m, &n);
  printf("Il massimo comun divisore di %d e %d e' ", m, n);
  while (m != n)
    if (m > n)
      m = m - n;
    else
      n = n - m;
  printf("%d\n", m);
  return 0;
}
```