Capitolo 3

Correttezza degli algoritmi

3.1 Algoritmi iterativi

Per dimostrare la correttezza degli algoritmi iterativi si utilizza l'*invariante di ciclo*. L'invariante è una proposizione che riguarda riguardante i contenuti delle variabili della procedura o del programma e che rispetta le seguenti proprietà:

Inizializzazione: la proposizione vale immediatamente prima di entrare nel ciclo.

Mantenimento: se la proposizione vale prima di eseguire un'iterazione, allora vale anche al termine dell'iterazione stessa.

Terminazione: al termine del ciclo (qualora il ciclo termini), la proposizione permette di ricavare la proprietà che permette di dimostrare la correttezza dell'algoritmo.

Per provare che il ciclo termina bisogna definire una funzione strettamente decrescente ad ogni iterazione e dimostrare che non può decrescere all'infinito.

Esempio 3.1. Consideriamo l'algoritmo `fibonacciIterativoModificato`, del quale vogliamo dimostrare il fatto che calcoli l'n-esimo numero di Fibonacci; l'invariante di ciclo è il seguente:

$$\text{Ad ogni iterazione, } b = F_{i-1} \text{ e } a = F_{i-2}$$

Inizializzazione: poiché $i = 3$, dobbiamo verificare che $b = F_2$ e $a = F_1$; la dimostrazione è immediata, poiché $b = 1 = F_2$ e $a = 1 = F_1$.

Mantenimento: assumiamo l'invariante verificato per una generica i-esima iterazione e dimostriamo che esso vale anche al termine di tale iterazione; ricordiamo inoltre, nonostante non sia esplicitamente indicato, che al termine dell'iterazione viene incrementato l'indice i. Grazie alla nostra assunzione, abbiamo $b = F_{i-1}$ e $a = F_{i-2}$: alla variabile c viene assegnato il valore della somma $a + b = F_{i-2} + F_{i-1} = F_i$, alla variabile a viene assegnato il valore di $b = F_{i-1}$, alla variabile b il valore di $c = F_i$ ed i viene incrementata ($i = i+1$); è immediato dimostrare che continua a valere l'invariante, ossia che $b = F_{(i+1)-1}$ e $a = F_{(i+1)-2}$.

Terminazione: all'uscita del ciclo, si ha $i = n+1$, dunque $b = F_{(n+1)-1} = F_n$, che è il valore restituito, come volevasi dimostrare.

3.2 Algoritmi ricorsivi

La dimostrazione viene svolta procedendo per *induzione*. Occorre formalizzare una proprietà utile per dimostrare la correttezza dell'algoritmo e provare che:

- valga per i *casi base*;
- assumendo che valga per problemi di dimensione inferiore, ossia per le chiamate ricorsive eseguite, provare che vale anche per il problema iniziale (*passo induttivo*).

Esempio 3.2. Consideriamo l'algoritmo `fibonacciRicorsivo` e formalizziamo la seguente proprietà:

L'output della chiamata di funzione `fibonacciRicorsivo`(n) è l'n-esimo numero di Fibonacci.

Casi base: per $n = 1$ e $n = 2$, l'algoritmo restituisce $1 = F_1 = F_2$.

Ipotesi induttiva: supponiamo che, per $k < n$, `fibonacciRicorsivo`(k) restituisca F_k.

Passo induttivo: l'algoritmo restituisce `fibonacciRicorsivo`$(n-1)$ + `fibonacciRicorsivo`$(n-2)$ (sia $n > 2$); per ipotesi, essi restituiscono, rispettivamente, F_{n-1} e F_{n-2}, la cui somma è F_n, come volevasi dimostrare.