Corso di Programmazione I

La ricorsione

Riferimenti

- Fondamenti di Informatica I
  - Parte II, capitolo IX: §1, §2, §4, §5, §6

  oppure (trattazione ridotta)
- Elementi di Informatica
  - Parte II, capitolo VII: §8

La ricorsione

Il concetto di ricorsione in informatica si riconduce a quello di induzione matematica

Sia P un predicato sull'insieme N dei numeri naturali e sia vero il predicato P(0); se per ogni *K intero,* dal predicato P(k) discende la verità di P(k+1) allora P(n) è varo per qualsiasi n.

Ricorsione

La dimostrazione in forma induttiva viene dunque svolta secondo i seguenti passi:

- Passo base dell'induzione: dimostrare P(0)=vero
- Passo induttivo: dimostrare che per ogni k>0

  P(k) ⟹ P(k+1)

Così come nel principio di induzione la verità di P(k+1) discende dalla verità dello stesso predicato P(k), il calcolo di un funzione ricorsiva avviene mediante il calcolo della stessa funzione in un passo successivo

Esempio

- **Si vuole dimostrare per induzione che:** $S(n) = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$
- **Base di induzione:** $S(0) = \sum_{i=0}^{0} 2^i = 2^1 - 1$
- **Ipotesi di induzione: per ogni n>0 e' vero che:** $S(n-1) = \sum_{i=0}^{n-1} 2^i = 2^n - 1$
- **Dimostrazione:**

$$S(n) = \sum_{i=0}^{n} 2^i = \sum_{i=0}^{n-1} 2^i + 2^n = 2^n - 1 + 2^n = 2 * 2^n - 1 = 2^{n+1} - 1$$

Ricorsione

Il calcolo ricorsivo consiste nel calcolare una funzione attraverso se stessa.

Esempio:
La funzione fattoriale *n!* (per interi non negativi

a) *0!* = 1
b) se *n>0* allora *n! = n*(n-1)!*

Ricorsione

In altre parole,
la potenza della ricorsione nasce dalla possibilità di definire un insieme infinito di oggetti con regola finita.

...allo stesso modo, un insieme infinito di computazioni può essere descritto con un programma ricorsivo finito.

Algoritmi ricorsivi

Sono caratterizzati da:

- Una funzione generatrice

$$x_{i+1} = g(x_i)$$

che produce una sequenza $x_1, x_2, \ldots, x_n$

- Un predicato p=p(x_i) vero per x=x_n

- un funzione f(x) detta ricorsiva tale che

$$\begin{cases} f(x_i) = c[h(x_i), f(x_{i+1})] \\ f(x_n) = \overline{f} \end{cases}$$

Algoritmi ricorsivi

Il calcolo della funzione ricorsiva quindi si riconduce a

-Generazione della sequenza $x_1, x_2, \ldots, x_n$

- Calcolo di $f(x_n) = \overline{f}$

- Calcolo di

$$f(x_i) = c[h(x_i), f(x_{i+1})] \quad per \quad i = n..1$$

Schema di algoritmi ricorsivi

Sono codificati mediante un sottoprogramma che richiama se stesso

```
T P (T1 x) {
     ...
     if p(x) F;
     else C(S,P);
     R
}
```

```
int fatt(int x){
     int f;
     if (x<=1) f=1;
     else f=x*fatt(x-1);
     return f;
}
```

Esempio:La funzione fattoriale *n!* (per interi non negativi)

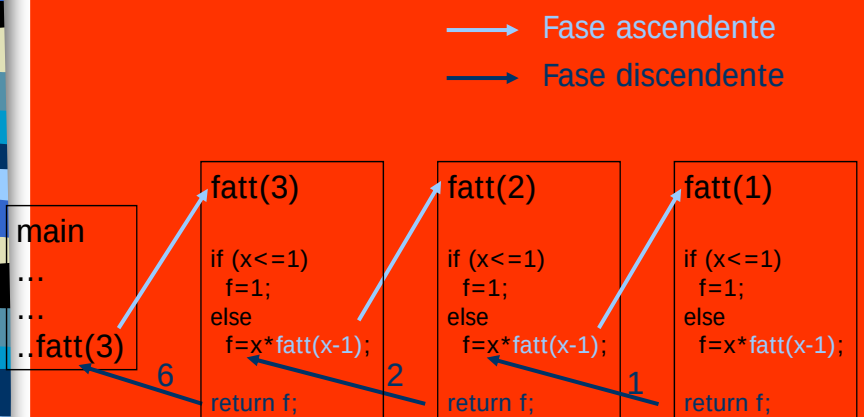
a) *0!* = 1

b) se *n>1* allora *n! = n*(n-1)!*

Terminazione

- La chiamata ricorsiva deve quindi essere subordinata ad una condizione che ad un certo istante risulti non soddisfatta (il predicato p)
- Il numero di chiamate necessarie viene detto *profondità* della ricorsione

Schema ricorsivo



Meccanismo interno di ricorsione

Un sottoprogramma ricorsivo è un sottoprogramma che richiama direttamente o indirettamente se stesso.

Non tutti i linguaggi realizzano il meccanismo della ricorsione. Quelli che lo realizzano possono utilizzare due tecniche:

- gestione LIFO di più copie della stessa funzione, ciascuna con il proprio insieme di var. locali
- Gestione mediante *record di attivazione*; un' unica copia del sottoprogramma ma ad ogni chiamata è associato un record di attivazione (var.locali e punto di ritorno).

Forma iterativa

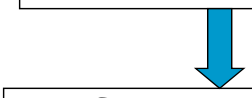
Un problema ricorsivo può essere risolto in termini iterativi.
Nel caso in cui la ricorsione è in coda, il problema è molto semplice

```
P (T1 x) {
      ...
      if p(x) F;
      else {S;P(x);}
      return;
}
```

```
while !p(x)
      S;
F;
```

Forma iterativa: Esempio

```
int fatt(int x){
 int f;
 if (x<=1) f=1;
 else f=x*fatt(x-1);
 return f;
}
```



```
int fatt(const int x){
 int f=1,i;
 for(i=1;i<=x;i++)
     f=f*i;
 return f;
}
```

Ricorsione e iterazione

- Uso di un costrutto di selezione
- Condizione di terminazione
- Non convergenza

- Uso di un costrutto di iterazione
- Condizione di terminazione
- Loop infinito

A differenza dell'iterazione, la ricorsione richiede un notevole overhead dovuto alle chiamate di funzione. Dato che un programma ricorsivo può essere sempre trasformato in un programma iterativo, perché usare la ricorsione?

Quando usare/non usare la ricorsione

- La ricorsione deve essere evitata quando esiste una soluzione iterativa ovvia e in situazioni in cui le prestazioni del sistema sono un elemento critico
- Algoritmi che per loro natura sono ricorsivi piuttosto che iterativi dovrebbero essere formulati con procedure ricorsive. Ad esempio considerare che alcune strutture dati sono inerentemente ricorsive:
  - Strutture ad albero
  - Sequenze
  - …..

Le torri di Hanoi

- **Soluzione ricorsiva**

Il problema con un solo disco è banale

sposto il disco dalla 1° alla 3° colonna

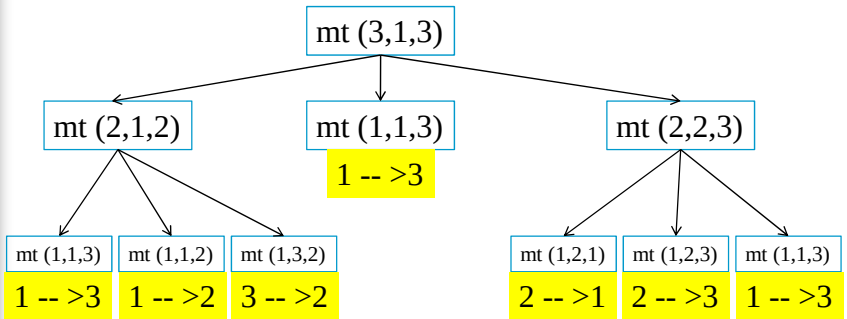
Se i dischi sono **N** e se sono riuscito a risolvere il problema con **N-1** dischi (induzione), allora

sposto N-1 dischi dalla 1° alla 2° colonna
sposto il primo disco dalla 1° alla 3° colonna
sposto N-1 dischi dalla 2° alla 3° colonna

Codifica

```
procedure Hanoi(N, Origine, Libero, Arrivo: Byte);
begin
   if(N = 1) then
      write(Origine:2, ' --> ', Arrivo:2)
   else
    begin
       muovitorre (altezza - 1,partenza,ausiliario);
       muovidisco (partenza,arrivo);
       muovitorre (altezza - 1,ausiliario,arrivo);
  end;
end;
```

Albero delle chiamate (N=3)



Complessità

$$
\begin{aligned}
T(n) &= 2\,T(n-1) + 1 \\
&= 2\,(2\,T(n-2) + 1) + 1 \\
&= 2^2\,T(n-2) + 2 + 1 \\
&= 2^2\,(2\,T(n-3) + 1) + 2 + 1 \\
&= 2^3\,T(n-3) + 2^2 + 2 + 1 \\
&= \ldots \\
&= 2^k\,T(n-k) + 2^{k-1} + \ldots + 2^2 + 2 + 1 \\
&= 2^k\,T(n-k) + 2^k - 1.
\end{aligned}
$$

La somma

$$2^{k-1} + \ldots + 2^2 + 2 + 1$$

é un caso particolare della **serie geometrica**

$$\sum_{i=0}^{m} q^i = \frac{q^{m+1} - 1}{q - 1}$$

per cui, ponendo $q = 2$ si ha $2^{k-1} + \ldots + 2^2 + 2 + 1 = 2^k - 1$. Per $k = n - 1$ si ha, ricordando che $T(1) = 1$

$$
\begin{aligned}
T(n) &= 2^k\,T(n-k) + 2^k - 1 \\
&= 2^{n-1}T(1) + 2^{n-1} - 1 \\
&= 2^n - 1.
\end{aligned}
$$