La ricorsione

Fulvio CORNO - Matteo SONZA REORDA
Dip. Automatica e Informatica
Politecnico di Torino

Sommario

- Definizione di ricorsione e strategie *divide et impera*
- Semplici algoritmi ricorsivi
- Merge Sort
- Quicksort
- Esempi più complessi di algoritmi ricorsivi

Definizione

- Una procedura si dice *ricorsiva* (o ricursiva) quando:
  - all'interno della propria definizione, compare la chiamata alla procedura stessa
  - oppure compare la chiamata ad almeno una procedura la quale, direttamente o indirettamente, chiama la procedura stessa.
- Un algoritmo si dice ricorsivo quando si basa su procedure ricorsive.

Esempio: il fattoriale

- $0! \equiv 1$
- $\forall N \geq 1$:
  $N! \equiv N \cdot (N-1)!$

```
double fact( double N )
{
    double sub ;
    if(N == 0)
        return 1.0 ;
    else
    {
        sub = fact( N-1 ) ;
        return N * sub ;
    }
}
```

Motivazioni

1.3

- Molti problemi si prestano, per loro natura, ad una descrizione ricorsiva:
  - Si definisce un metodo per risolvere dei sotto-problemi analoghi a quello di partenza (ma più piccoli)
  - Si definisce un metodo per combinare le soluzioni parziali nella soluzione del problema originario.

Divide et Impera (I)

"a" sottoproblemi, ciascuno "b" volte più piccolo del problema

- Soluzione = Risolvi(Problema)
- Risolvi(Problema):
  - $Sottoproblema_{1,2,3,\ldots,a}$ = Dividi(Problema) ;
  - Per ciascun $Sottoproblema_i$:
    - $Sottosoluzione_i$ = Risolvi($Sottoproblema_i$) ;
  - Return Soluzione = Combina($Sottosoluzione_{1,2,3,\ldots,a}$) ;

Divide et Impera (II)

- Soluzione = Risolvi(Problema) ;
- Risolvi(Problema):
  - $\text{Sottoproblema}_{1,2,3,\ldots,a}$ = Dividi(Problema) ;
  - Per ciascun Sottoproblema_i:
    - Sottosoluzione_i = Risolvi(Sottoproblema_i) ;
  - Return Soluzione = Combina($\text{Sottosoluzione}_{1,2,3,\ldots,a}$) ;

Chiamata ricorsiva

7 a.a. 2001/2002

Quando fermarsi?

La ricorsione non deve essere infinita, in quanto ogni algoritmo deve terminare.

Ad un certo punto, i sottoproblemi diventano così semplici da essere risolvibili:

- in modo banale (es.: insiemi di 1 solo elemento)
- oppure, con metodi alternativi alla ricorsione.

8 a.a. 2001/2002

Avvertenze

- Ricordare sempre la condizione di terminazione
- Fare in modo che i sottoproblemi siano tutti strettamente “minori” del problema iniziale

Divide et Impera (completo di terminazione)

Risolvi(Problema):

- Se il problema è banale:
  - Soluzione = Risolvi_banale(Problema)
- Else:
  - $\text{Sottoproblema}_{1,2,3,\ldots,a}$ = Dividi(Problema) ;
  - Per ciascun Sottoproblema_i:
    - Sottosoluzione_i = Risolvi(Sottoproblema_i) ;
  - Return Soluzione = Combina($\text{Sottosoluzione}_{1,2,3,\ldots,a}$) ;

Complessità (I)

T(n)

Risolvi(Problema):

- Se il problema è banale (n≤c):
  - Soluzione = Risolvi_banale(Problema)

Θ(1)

- Else:
  - $Sottoproblema_{1,2,3,...,a}$ = Dividi(Problema) ;

D(n)

  - Per ciascun $Sottoproblema_i$:
    - $Sottosoluzione_i$ = Risolvi($Sottoproblema_i$) ;

T(n/b)

  - Return Soluzione = Combina($Sottosoluzione_{1,2,3,...,a}$) ;

C(n)

Complessità (II)

T(n) =

- Θ(1) per n ≤ c
- D(n) + a T(n/b) + C(n) per n > c

Equazione alle ricorrenze di non facile soluzione.

Se D(n)+C(n)=Θ(n), si trova T(n)=Θ(n log n).

Sommario

- Definizione di ricorsione e strategie *divide et impera*
- Semplici algoritmi ricorsivi
- Merge Sort
- Quicksort
- Esempi più complessi di algoritmi ricorsivi

Esempio: numeri di Fibonacci

Problema:

- Calcolare l'N-esimo numero di Fibonacci

Definizione:

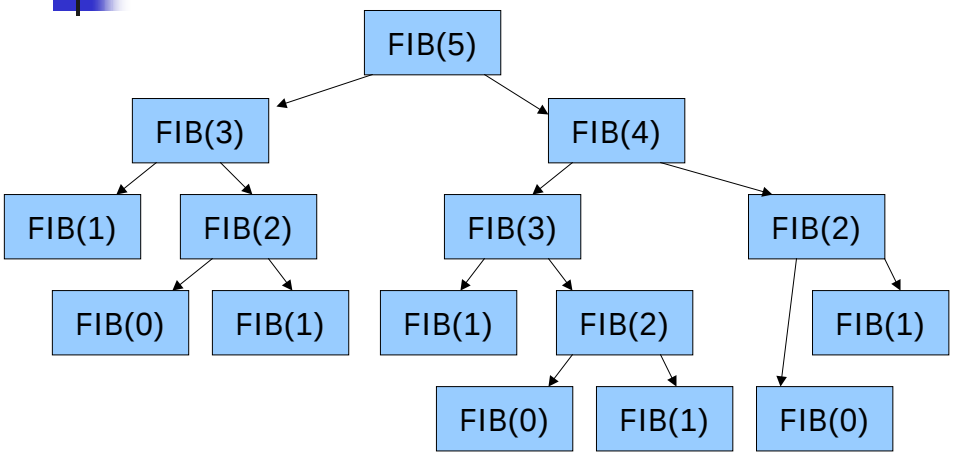
- $FIB_{N+1} = FIB_N + FIB_{N-1}$ per n>0
- $FIB_1 = 1$
- $FIB_0 = 0$

Soluzione

```
long fib(int N)
{
     long f1, f2 ;

     if(N == 0) return 0 ;
     if(N == 1) return 1 ;
     else
     {
          f1 = fib(N-1) ;
          f2 = fib(N-2) ;
          return f1+f2 ;
     }
}
```


Analisi



Esempio: ricerca dicotomica

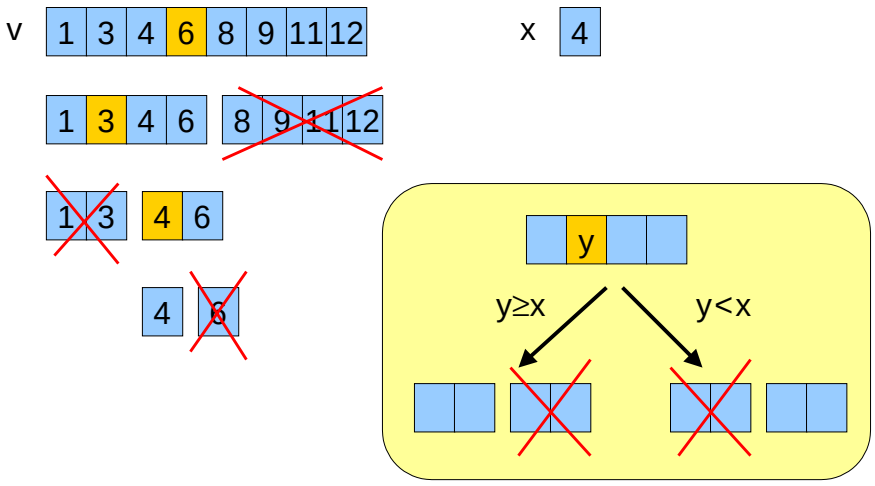
Problema

- Determinare se un elemento x è presente all'interno di un vettore ordinato v[N]

Approccio

- Dividere il vettore a metà, e riapplicare il problema su una delle due metà (l'altra si può escludere a priori, essendo il vettore ordinato)

Esempio



Soluzione

```
int trova(int v[], int a, int b, int x)
{
     int c ;
     if(b-a == 0)
          if(v[a]==x) return a ;
          else return –1 ;
     else
     {
          c = (a+b) / 2 ;
          if(v[c] >= x)
               return trova(v, a, c, x) ;
          else return trova(v, c, b, x) ;
     }
}
```


Esercizio proposto

Si calcoli il coefficiente binomiale (n m), sfruttando le relazioni (derivate dal Triangolo di Tartaglia):

$$\begin{cases} \binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m} \\ \binom{n}{n} = \binom{n}{1} = 1 \\ 1 \le n, \quad 1 \le m \le n \end{cases}$$

Esercizio proposto

Si esegua il calcolo del determinante di una matrice quadrata.

Si ricorda che:

- $Det(M_{1x1}) = m_{11}$
- $Det(M_{NxN})$ = somma dei prodotti degli elementi di una riga (o colonna) per i determinanti delle sottomatrici (N-1)x(N-1) ottenute cancellando la riga e la colonna che contengono l'elemento, preso con segno $(-1)^{i+j}$.

Ricorsione ed iterazione

Ogni programma ricorsivo può anche essere implementato in modo iterativo.

La soluzione migliore, come efficienza e chiarezza del codice, dipende dal problema.

Esempio: il fattoriale (iterativo)

- $0! \equiv 1$
- $\forall N \geq 1$:
  $N! \equiv N \cdot (N-1)!$

```
double fact( double N )
{
    double tot = 1.0 ;
    int i;
    for(i=2; i<=N; ++i)
        tot = tot * i ;
    return tot ;
}
```


Fibonacci (iterativo)

```
long fib(int N)
{    long f1p=1, f2p=0, f=f1p+f2p ;
     int i;
     if(N == 0) return 0 ;
     if(N == 1) return 1 ;
     else
     {    for(i=2; i<= N; ++i)
          {    f2p = f1p ;
               f1p = f ;
               f = f1p+f2p ;
          }
          return f ;
     }
}
```

Ricerca dicotomica (iterativo)

```
int trova(int v[], int a, int b, int x)
{
     int c ;
     while(b-a != 0)
     {
          c = (a+b) / 2 ;
          if(v[c] >= x)
               b = c ;
          else a = c ;
     }
     if(v[a]==x) return a ;
     else return –1 ;
}
```


Esercizi proposti

1. Si fornisca la versione iterativa del calcolo del coefficiente binomiale (n m).

2. Si analizzino le difficoltà nel realizzare la versione iterativa del calcolo del determinante.

Sommario

- Definizione di ricorsione e strategie *divide et impera*
- Semplici algoritmi ricorsivi
- Merge Sort
- Quicksort
- Esempi più complessi di algoritmi ricorsivi

Pseudo-codice

```
MERGE-SORT(A, p, r)
1  if p < r
2     then q ← ⌊(p + r)/2⌋
3          MERGE-SORT(A, p, q)
4          MERGE-SORT(A, q + 1, r)
5          MERGE(A, p, q, r)
```

Sommario

- Definizione di ricorsione e strategie *divide et impera*
- Semplici algoritmi ricorsivi
- Merge Sort
- Quicksort
- Esempi più complessi di algoritmi ricorsivi

Sommario

- Definizione di ricorsione e strategie *divide et impera*
- Semplici algoritmi ricorsivi
- Merge Sort
- Quicksort
- Esempi più complessi di algoritmi ricorsivi