Michele Scquizzato

SCQUIZZA@MATH.UNIPD.IT

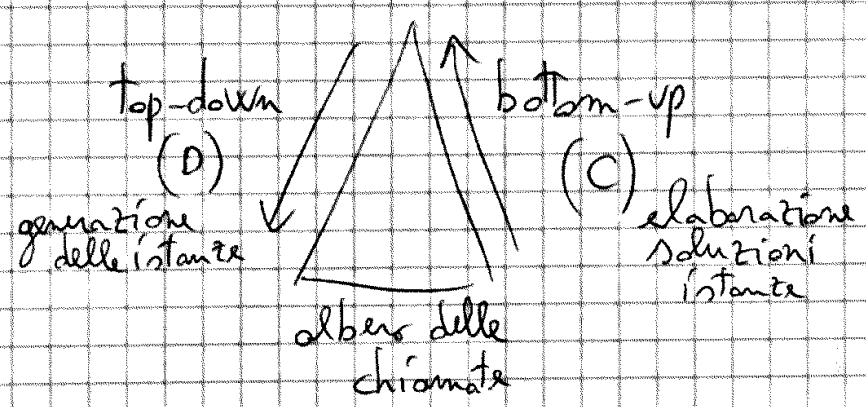
WWW.MATH.UNIPD.IT/~SCQUIZZA

ricevimento: venerdì 17:00 – 18:00 su appuntamento, Zoom

Programmazione Dinamica

2° paradigma di programmazione (dopo il D&C)

critica al Divide & Conquer:



il processo di soluzione non ha memoria ⇒ se una sottoistanza deve essere ricombinata in un altro punto dell'albero va ricalcolata

problema: sottoistanze ripetute

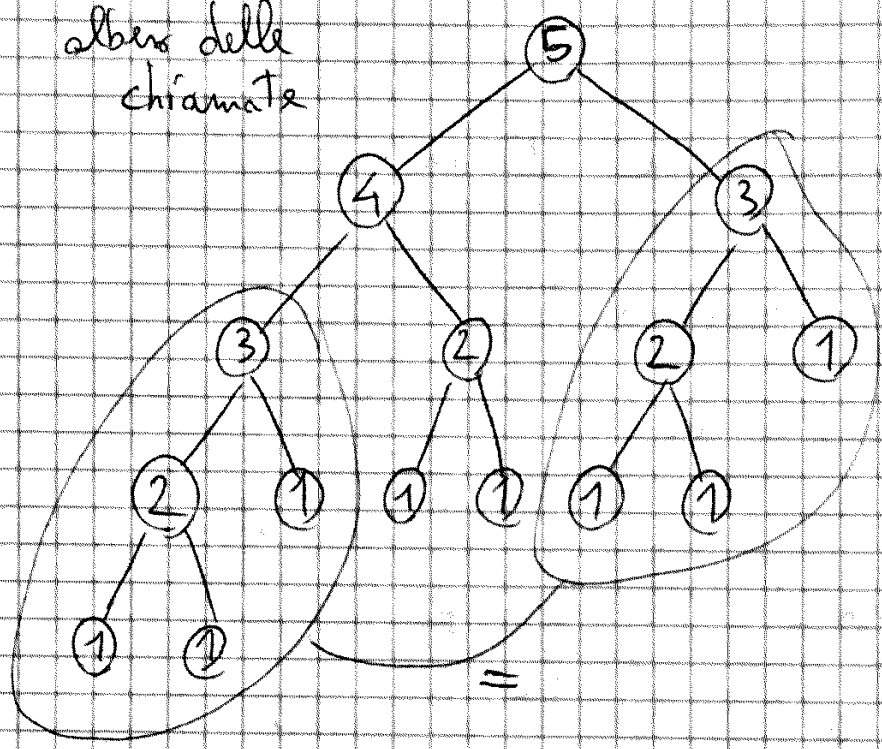
Esempio: sequenza di Fibonacci: 1, 1, 2, 3, 5, 8, 13, ...

$$F_n = \begin{cases} 1 & n = 0,1 \\ F_{n-1} + F_{n-2} & n \geq 2 \end{cases}$$

Algoritmo D&C:

```
REC_FIB (n)
    if (n=0) or (n=1) then return 1
    return REC_FIB(n-1) + REC_FIB(n-2)
```

albero delle chiamate



Complessità:

$$T_F(n) = \begin{cases} 0 & n=0,1 \\ T_F(n-1) + T_F(n-2) + 1 & n \geq 2 \end{cases}$$

difficile da risolvere esattamente, ma a noi basta dire $T_F(n) \geq f(n)$

unfolding:

se $n \geq 2$ $T_F(n) = T_F(n-1) + T_F(n-2) + 1 \geq 2T_F(n-2) + 1$

$$\geq 2\left(2T_F(n-2-2)+1\right)+1 = 2^2 T_F(n-2\cdot 2)+2+1$$

$$\geq \cdots$$

$$\vdots$$

$$\geq 2^i T_F(n-2i) + \sum_{j=0}^{i-1} 2^j$$

$\downarrow$

caso base per $i = \left\lfloor \frac{n}{2} \right\rfloor$

n pari: $2^{\frac{n}{2}} T_F\left(n - 2\frac{n}{2}\right) + \sum_{j=0}^{\frac{n}{2}-1} 2^j$

n dispari: $\left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$

$$2^{\frac{n-1}{2}} T_F\left(n - 2 \cdot \frac{n-1}{2}\right) + \sum_{j=0}^{\frac{n-1}{2}-1} 2^j$$

$$T_F \geq \sum_{j=0}^{\left\lfloor \frac{n}{2} \right\rfloor - 1} 2^j = \frac{2^{\left\lfloor \frac{n}{2} \right\rfloor} - 1}{2-1} = \Theta\left(2^{\frac{n}{2}}\right)$$

$$\Rightarrow T_F = \Omega\left(2^{\frac{n}{2}}\right)$$

cioè <u>esponenziale</u>!

cosa significa: che il numero di sottoistanze ripetute è elevatissimo

$\hookrightarrow$ e ripetizioni di una certa sottoistanza

algoritmo iterativo:

```
IT_FIB (n)
  if (n=0) or (n=1) then return 1
  F[0] ← 1                    \\ F = tabella (anzi, array) di n elementi
  F[1] ← 1
```

$\downarrow$

```
for i = 2 to n do
    F[i] ← F[i-1] + F[i-2]
return F[n]
```

Complessità: $\Theta(n)$

Cosa fa questo alg.: memorizza in una struttura dati ausiliaria tutte le soluzioni alle sottoistanze $\Rightarrow$ evito di ricalcolarle

Nota: IT_FIB salta completamente la fase top-down.
È possibile mantenere ~~&~~ i vantaggi del top-down, e quindi del D&C, senza incorrere nel problema del ricalcolo di soluzioni?

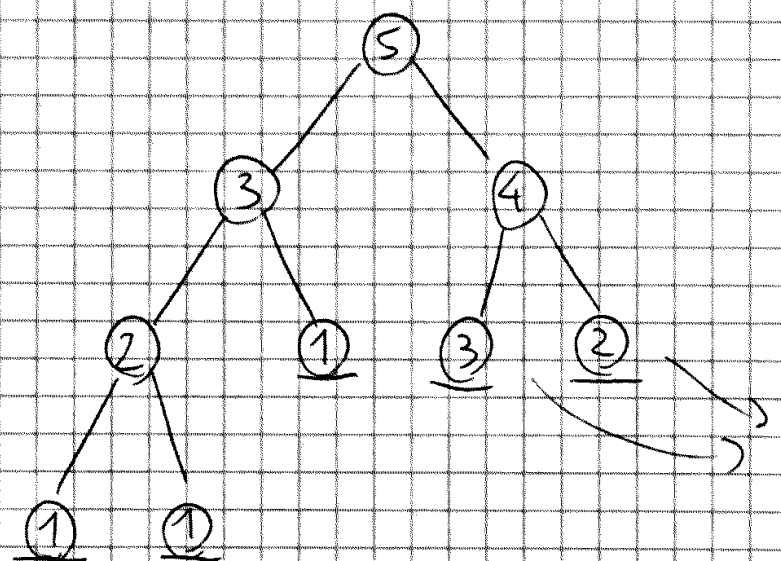
MEMOIZZAZIONE

Definizione: un algoritmo memoizzato è costituito da 2 subroutine
- <u>routine di inizializzazione</u>: risolve direttamente i casi base e inizializza una struttura dati ("tabella") che contiene le soluzioni ai casi base e elementi per tutte le sottoistanze da calcolare inizializzate a un valore di default. Infine, invoca la routine ricorsiva
- <u>routine ricorsiva</u>: segue il codice del D&C preceduto da un test sulla tabella per verificare se la soluzione è già stata calcolata e memorizzata. Se sì, ritorna, se no la si calcola ricorsivamente e la si memorizza in tabella

```
INIT_FIB (n)
    if (n=0) or (n=1) then return 1
    F[0] ← 1
    F[1] ← 1
    for i=2 to n do
        F[i] ← 0
    return REC_FIB (n)
```

valore di default

```
REC_FIB (i)
    if F[i] = 0 then
        F[i] ← REC_FIB (i-2) + REC_FIB (i-1)
    return F[i]
```



non espansi perché già calcolati

$$n^\circ \text{ nodi} = \underbrace{n-1}_{\text{interni}} + \underbrace{n}_{\text{foglie}}$$

Complessità: $\Theta(n)$