Lezione 24 aprile

Friday, April 24, 2020 10:59 AM

Programmazione dinamica

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_i = F_{i-1} + F_{i-2}, \quad i > 1 \end{cases}$$

formula di Binet

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right)$$

```
Fib(n):
if (n <= 1) return n;
    return Fib(n-1) + Fib(n-2);
```

È un buon algoritmo?

$$T(n) = T(n-1) + T(n-2) + c \geq 2T(n-2) + c$$

$$= 2\left(2T\left(\frac{n}{4}\right) + c\right) + c$$

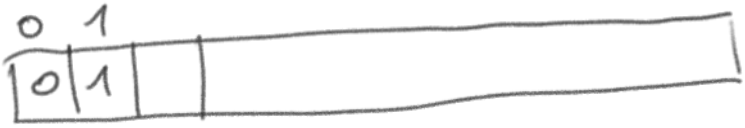
$$= 2^i T(1) + \overbrace{c + \cdots + c}^{i} \quad = i = n/2$$

$T(n) \geq 2^{n/2}$ esponenziale in n

Memorizzare le soluzioni parziali

Fib2 (n); usa tabella Fib

```
Fib[0] = 0;
Fib[1] = 1;
for (i = 2; i ≤ n; i++)
    Fib[i] = Fib[i-1] + Fib[i-2];
```



Fib2 è lineare in n spazio $\Theta(n)$

Fib era esponenziale in n

```
Fib3 (n):
if (n ≤ 1) return n;
a = 0; b = 1;
for (i = 2; i ≤ n; i++) {
    c = a + b;
    a = b;
    b = c;
} return c;
```

Fib3 : $\Theta(n)$ tempo
$\Theta(n)$ spazio

Fib ~~esponenziale~~ in n.

I numeri di Fibonacci erano esp.

Fib3 (n) $\Theta(n)$

dimensione del problema? $\log n$

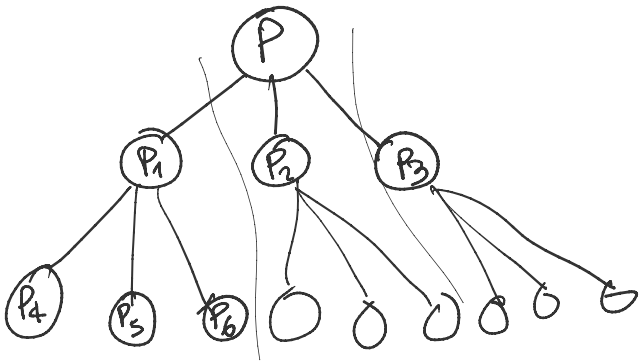
Fib3 (1000000)

è esponenziale in $\log n$

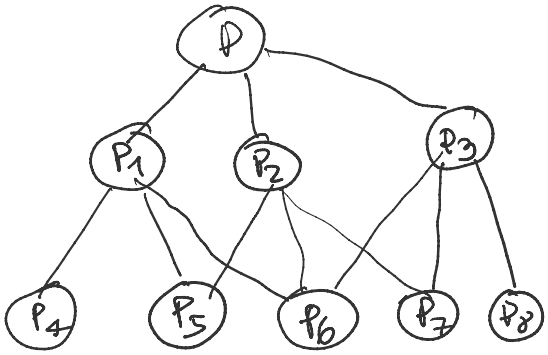
Fib è doppiamente esponenziale!!

Programmazione dinamica

Risoluzione per sottoproblemi



D.I



P.D.

Uno stesso sottoproblema può essere utilizzato più volte

P.D. si applica in genere a problemi

v. d. si applica ... di ottimizzazione (min, max).

Si applica quando la soluzione ottima si può ricavare dalle soluzioni ottime dei sottoproblemi.

Un algoritmo di programmazione dinamica viene definito in 4 passi:

1) Problema generale e sottoproblemi si definiscono nello stesso modo. $F_n \dots F_i$
2) Definizione dei sottoproblemi elementari $F_0 = 0$ e $F_1 = 1$
3) Regola ricorsiva di calcolo
4) Memorizzazione e ordinamento in tabella dei sottoproblemi. Fib

LCS: Longest common Subsequence
Sequenza comune di lunghezza massima

b: D A A B D C D A A C A C B A, n
a: A D C A A B, m

2 sequenze

Problema molto importante

Problema molto importante

SICUREZZA

BIOINFORMATICA

DNA stringhe di caratteri A T C G

$b: b_1 \dots b_n$ $\quad b[1,j]$ prefissi

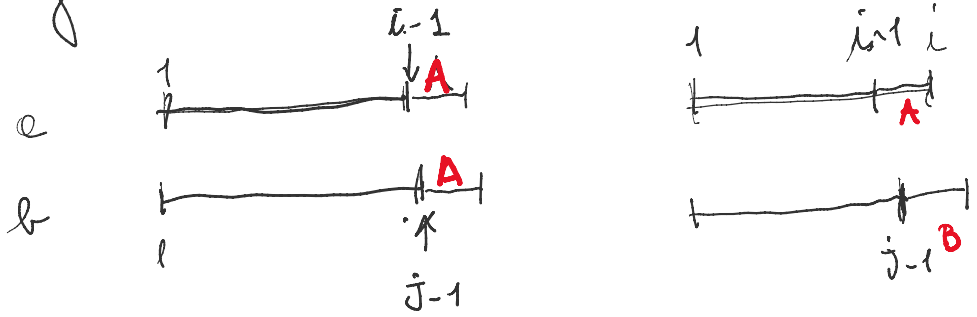
$a: a_1 \dots a_m$ $\quad a[1,i]$

1) $LCS(a[1,j], b[1,j]) = L[i,j]$ sottoproblemi

$LCS(a[1,m], b[1,n]) = L[m,n]$ problema

2) $L(\phi, j) = 0 \qquad L(i, \phi) = 0$

3) regola ricorsiva



$$L[i,j] = \begin{cases} \phi \text{ se } i=0 \text{ o } j=0; \\ L[i-1, j-1] + 1 \text{ se } a_i = b_j \\ \max\left(L[i-1, j], L[i, j-1]\right) \end{cases}$$

4) Uso una tabella L (array a 2 dimensioni)

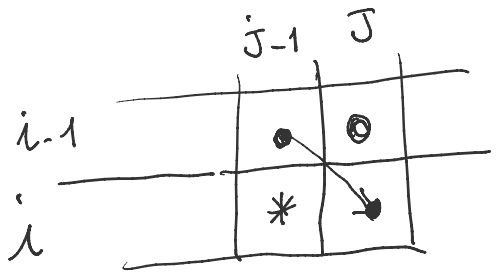
$[0..m, 0..n]$

a = AMICA

b = MATEMATICA

	∅	M	A	T	E	M	A	T	I	C	A
∅	∅	0	∅	∅	∅	∅	∅	∅	∅	∅	∅
A	∅	∅	1	1	1	1	1	1	1	1	1
M	∅	1	1	1	1	2	2	2	2	2	2
I	∅	1	1	1	1	2	2	2	3	3	3
C	∅	1	1	1	1	2	2	2	3	4	4
A	∅	1	2	2	2	2	3	3	3	4	5

$(m+1) \times (n+1)$



procediamo per righe

MATCHING → caratteri uguali

MISMATCH → caratteri diversi

LCS [m,n] dà la risposta al problema

LCS [m,n] = 5

```
LCS (a, b):
Θ(m)   for (i = 0; i ≤ m; i++)
         L[i,0] = 0;
Θ(n)   for (j = 0; j ≤ n; j++) L[0,j] = 0;
       for (i = 1; i ≤ m; i++)
Θ(n.m)   for (j = 1; j ≤ n; j++){
           if (a[i] == b[j])
             L[i,j] = L[i-1,j-1]+1;
```

```
if (...)
    L[i,j] = L[i-1,j-1]+1;
else if (L[i-1,j] > L[i,j-1]
    L[i,j] = L[i-1,j];
else L[i,j] = L[i,j-1];
}
```

Complessità: tempo $\Theta(n \cdot m)$

spazio $\Theta(n \cdot m)$?

È necessario lo spazio dell'array?

Se voglio calcolare solo la LCS basta $\Theta(n)$ spazio!

Se voglio la soluzione (cioè la seq. LCS) allora $\Theta(n \cdot m)$ spazio.