PROGRAMMAZIONE DINAMICA

- 1) caratterizzazione struttura di soluzione
- 2) definizione ricorsiva
- 3) eliminare le ripetizioni tramite memoization
- 4) Suiluppo di approccio bottom-up (iterativo)

Esempi:

- Fibonacci

RICORSIVO

```
\begin{split} &\operatorname{Fib}(n) \\ &\triangleright \operatorname{Pre:} \ n > 0 \ \operatorname{intero} \\ &\triangleright \operatorname{Post:} \ \operatorname{ritorna} \ \operatorname{l'} n\text{-mo numero della sequenza di Fibonacci} \\ &\mathbf{if} \ n \leq 2 \ \mathbf{then} \\ & f \leftarrow 1 \\ &\mathbf{else} \\ & f \leftarrow \operatorname{Fib}(n-1) + \operatorname{Fib}(n-2) \\ &\mathbf{end if} \\ &\mathbf{return} \ f \end{split}
```

complessita almeno

CON TEKOIZATION

```
FIB-MEMOIZATION(n, memo)

Pre: n > 0 intero, memo array di dim. > n

Post: ritorna F_n = n-mo numero della sequenza di Fibonacci if memo[n] \neq nil then

return memo[n]

end if posterior memo[n] non contiene alcun valore if n \leq 2 then

posterior memo[n] = memo[n] else

posterior memo[n] = memo[n] = memo[n]

else

posterior memo[n] = memo[n] = memo[n]

end if posterior memo[n] = memo[n]

return posterior memo[n]
```

Spazio usato per memo: $\Theta(n)$ Spazio usato per memo: $\Theta(n)$

VERSIONE BOTTOM-UP CON ARRAY

```
FIB-BOTTOMUP(n)

Pre: n > 0 intero

Post: ritorna F_n = n-mo numero della sequenza di Fibonacci if n \le 2 then

return 1

else

FIB[1..n] sia un array di dimensione n

FIB[1] \leftarrow 1, FIB[2] \leftarrow 1

for i \leftarrow 3 to n do \triangleright inv: \forall j < i. FIB[j] = F_j

FIB[i] \leftarrow FIB[i] \leftarrow FIB[i] + FIB[i] = F_j

end for
end if

return FIB[n]
```

Tempo e Spazio O(n)

VERSIONE BOTTOM-UP SENZA ARRAY

```
\begin{split} & \text{Fib-Iter}(n) \\ & \Rightarrow \text{Pre: } n > 0 \text{ intero} \\ & \Rightarrow \text{Post: ritorna } F_n = n\text{-mo numero della sequenza di Fibonacci} \\ & \text{if } n \leq 2 \text{ then} \\ & \text{return 1} \\ & \text{else} \\ & \text{FibA} \leftarrow 1, \text{FibB} \leftarrow 1 \\ & \text{for } i \leftarrow 3 \text{ to } n \text{ do} \\ & \Rightarrow \text{inv: FibA} = F_{i-1}, \text{FibB} = F_{i-2} \\ & tmp \leftarrow \text{FibA} + \text{FibB} \\ & \text{FibB} \leftarrow \text{FibA} \\ & \text{FibA} \leftarrow tmp \\ & \text{end for} \\ & \text{end if} \\ & \text{return FibA} \end{split}
```

Tempo: O(n)

Spazio: $\Theta(1) \rightarrow \text{tengo memoria solo di}$ FIB[i-2] e

FIB[i-1]

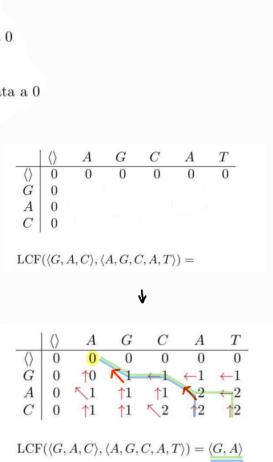
- Massima sottosequenza comune - LCS

```
date due sequenze S_1 e S_2:
S_1 = \text{ACCGGTCGAGTGCGCGGAAGCCGGCCGAA}
S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA}
la massima sottosequenza comune è S_3
S_1 = \text{ACCGGTCGAGTGCGCGGAAGCCGGCCGAA}
S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA}
S_3 = \text{GTCGTCGGAAGCCGGCCGAA}
```

| S3 | mi dà la misura di quanto si assomigliano S1 e S2

ALGORITHO BOTTOM-UP

```
LCS-BOTTOM-UP(X, Y)
\triangleright Pre: X = \langle x_1, \dots x_m \rangle, Y = \langle y_1, \dots, y_n \rangle
\triangleright Post: ritorna le matrici c[0..m, 0..n] e b[1..m, 1..n]
   m \leftarrow X.length, n \leftarrow Y.length
   siano c[0..m, 0..n] e b[1..m, 1..n] due nuove tabelle
                                ⊳ la prima riga è inizializzata a 0
   for j \leftarrow 0 to n do
       c[0,j] \leftarrow 0
   end for
   for i \leftarrow 0 to m do
                                  ⊳ la prima colonna è inizializzata a 0
       c[i,0] \leftarrow 0
   end for
  for i \leftarrow 1 to m do
       for j \leftarrow 1 to n do
            if x_i = y_i then
                 c[i,j] \leftarrow c[i-1,j-1] + 1
                 b[i,j] \leftarrow 
            else
                        \triangleright x_i \neq y_i
                 if c[i-1,j] \ge c[i,j-1] then
                      c[i,j] \leftarrow c[i-1,j]
                      b[i,j] \leftarrow \uparrow
                 else \triangleright c[i-1,j] < c[i,j-1]
                      c[i,j] \leftarrow c[i,j-1]
                      b[i,j] \leftarrow \leftarrow
                 end if
            end if
       end for
  end for
  return c, b
```



Complessita O(m·n)