PROGRAMMAZIONE DINAMICA

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

Esempi:

- Fibonacci

RICORSIVO

FIB(n)
▷ Pre: $n > 0$ intero
▷ Post: ritorna l'n-mo numero della sequenza di Fibonacci

```
if n ≤ 2 then
    f ← 1
else
    f ← FIB(n − 1) + FIB(n − 2)
end if
return f
```

complessita' almeno esponenziale

CON MEMOIZATION

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] ≠ nil then
    return memo[n]
end if        ▷ memo[n] non contiene alcun valore
if n ≤ 2 then
    f ← 1
else
    f ← FIB-MEMOIZATION(n−1, memo) + FIB-MEMOIZATION(n−2, memo)
end if
memo[n] ← f
return f
```

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

Complessita' temporale $\Theta(n)$

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 ≤ 2 then
    return 1
else
    FIB[1..n] sia un array di dimensione n
    FIB[1] ← 1, FIB[2] ← 1
    for i ← 3 to n do        ▷ inv: ∀j < i. FIB[j] = F_j
        FIB[i] ← FIB[i − 1] + FIB[i − 2]
    end for
end if
return FIB[n]
```

Tempo e Spazio $\Theta(n)$

VERSIONE BOTTOM-UP SENZA ARRAY

FIB-ITER(n)
▷ Pre: $n > 0$ intero
▷ Post: ritorna F_n = n-mo numero della sequenza di Fibonacci

```
if n ≤ 2 then
    return 1
else
    FIBA ← 1, FIBB ← 1
    for i ← 3 to n do        ▷ inv: FIBA = F_{i−1}, FIBB = F_{i−2}
        tmp ← FIBA + FIBB
        FIBB ← FIBA
        FIBA ← tmp
    end for
end if
return FIBA
```

Tempo: $\Theta(n)$

Spazio: $\Theta(1)$ → 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 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA

S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA

la massima sottosequenza comune è S_3

S_1 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA

S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA

S_3 = GTCGTCGGAAGCCGGCCGAA

$|S_3|$ mi dà la misura di quanto si assomigliano S_1 e S_2

ALGORITMO BOTTOM-UP

LCS-Bottom-Up(X, Y)
▷ Pre: $X = \langle x_1, \dots x_m \rangle$, $Y = \langle y_1, \dots, y_n \rangle$
▷ 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
for $j \leftarrow 0$ **to** n **do** ▷ la prima riga è inizializzata a 0
  $c[0, j] \leftarrow 0$
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 $x_i = y_j$ **then**
      $c[i, j] \leftarrow c[i-1, j-1] + 1$
      $b[i, j] \leftarrow \nwarrow$
    else ▷ $x_i \neq y_j$
      if $c[i-1, j] \geq c[i, j-1]$ **then**
        $c[i, j] \leftarrow c[i-1, j]$
        $b[i, j] \leftarrow \uparrow$
      else ▷ $c[i-1, j] < c[i, j-1]$
        $c[i, j] \leftarrow c[i, j-1]$
        $b[i, j] \leftarrow \leftarrow$
      end if
    end if
  end for
end for
return c, b

	$\langle\rangle$	A	G	C	A	T
$\langle\rangle$	0	0	0	0	0	0
G	0					
A	0					
C	0					

LCF$(\langle G, A, C \rangle, \langle A, G, C, A, T \rangle) =$

↓

	$\langle\rangle$	A	G	C	A	T
$\langle\rangle$	0	0	0	0	0	0
G	0	↑0	↖1	←1	←1	←1
A	0	↖1	↑1	↑1	↖2	←2
C	0	↑1	↑1	↖2	↑2	↑2

LCF$(\langle G, A, C \rangle, \langle A, G, C, A, T \rangle) = \langle G, A \rangle$

Complessità $\Theta(m \cdot n)$