b) $Z_{k-1} = \langle X_{i_1}, X_{i_2}, \dots, X_{i_{k-1}} \rangle$

$= \langle Y_{j_1}, Y_{j_2}, \dots, Y_{j_{k-1}} \rangle$

$i_{k-1} \leq i-1$
$j_{k-1} \leq j-1$ $\Rightarrow$ Z_{k-1} è sottosequenza di X_{i-1} e Y_{j-1}

ora dimostro che $Z_{k-1} = LCS(X_{i-1}, Y_{j-1})$

suppongo non vero, per assurdo: allora $\exists$ un'altra di lunghezza $\geq k$; a questa aggiungo in coda x_i, ottenendo una $CS(X_i, Y_j)$ di lunghezza $\geq k+1$: assurdo.

2) $x_i \neq y_j$ $(i, j > 0)$

basta dimostrare che $Z = LCS(X_i, Y_{j-1})$ oppure $Z = LCS(X_{i-1}, Y_j)$

a) $i_k = i$

$\Rightarrow j_k < j \Rightarrow Z$ è $SC(X_i, Y_{j-1})$

$x_i \neq y_j$ Z è anche $LCS(X_i, Y_{j-1})$ $\rightarrow$ per assurdo, come prima

b) $i_k < i$

$\Rightarrow Z$ è $SC(X_{i-1}, Y_j)$

Z è anche $LCS(X_{i-1}, Y_j)$ $\rightarrow$ per assurdo, come prima

Fine dim.

Passo 2: ricorrenza sui costi

chiamo $\ell(i,j) = |LCS(X_i, Y_j)|$

$\hookrightarrow$ funz. di costo

$$\ell(i,j) = \begin{cases} 0 & i=0 \text{ o } j=0 \quad 0) \\ 1+\ell(i-1,j-1) & i,j>0 \text{ e } x_i = y_j \quad 1) \\ \max\{\ell(i,j-1), \ell(i-1,j)\} & i,j>0 \text{ e } x_i \neq y_j \quad 2) \end{cases}$$

Per calcolare l'elemento L(i, j), bisogna già aver calcolato L(i, j-1) dando un ordine di calcolo delle soluzioni.

Genero due sottoistanze ricorsivamente; a differenza ad esempio di merge sort che crea due figli di ogni nodo, ne creo uno solo per ottimizzazione della prog. dinamica.

ci interessa $\ell(m,n)$

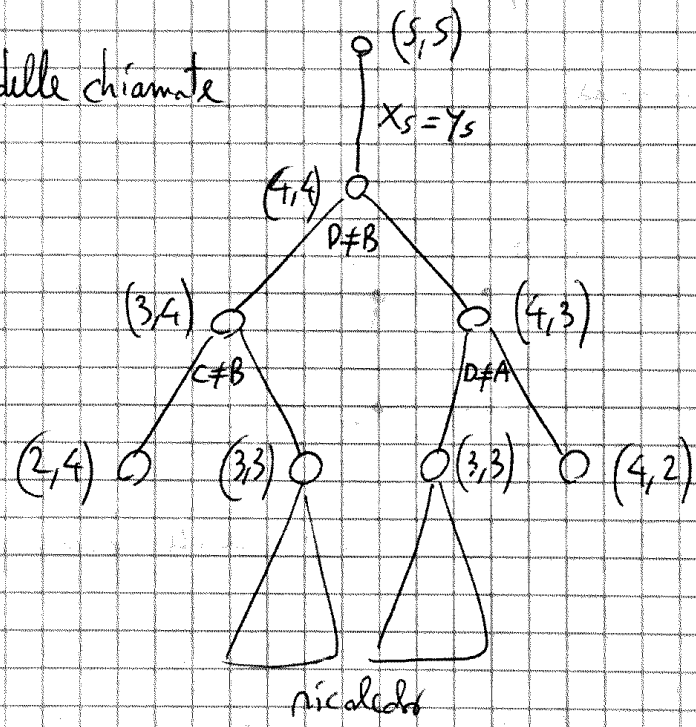
È chiara la presenza di sottoproblemi ripetuti: esempio:

$X = \langle A, B, C, D, E \rangle$

$Y = \langle B, C, A, B, E \rangle$

trova $\ell(5,5)$

albero delle chiamate



Modello di costo: confronti tra caratteri delle stringhe

$$T(m,n) = \begin{cases} 0 & m=0 \text{ o } n=0 \\ T(m-1,n) + T(m,n-1) + 1 & m,n>0 \end{cases}$$

↳ confronto x_i e y_j

Si può dimostrare che $T(m,n) = \Theta\binom{m}{n}$ (quando $m \geq n$)

vale che $\binom{m}{n} \geq \left(\frac{m}{n}\right)^n$

Nel caso pessimo, potrebbe diventare esponenziale, questo nel caso ricorsivo.

$m = 2n \Rightarrow T(2n, n) = \Omega(2^n)$ esponenziale nella taglia $(= 3n)$

$\Rightarrow$ D&C non funziona

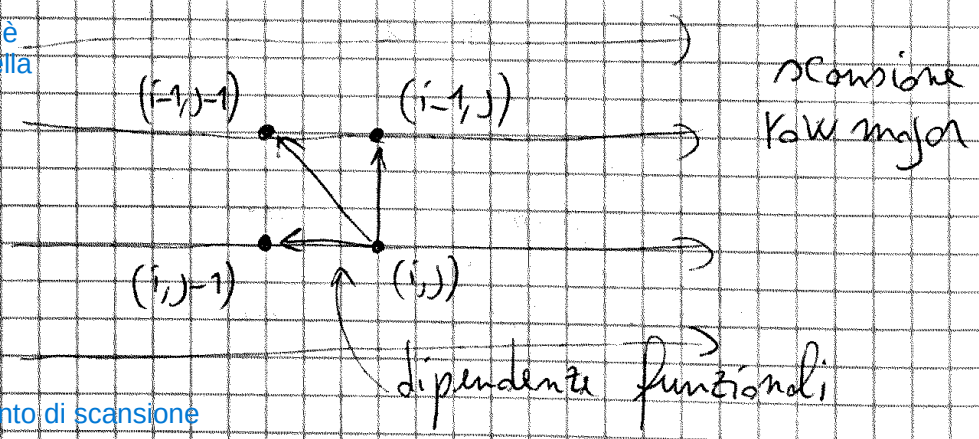
$\Rightarrow$ scrivo codice di prog. dinamica

bottom-up o memoizzata (o "iterativo" sinonimo)

$L[i,j] \leftarrow \ell(i,j)$

tabella bidimensionale

La scansione qui adoperata è a righe, altrimenti usano quella a colonne (column major).



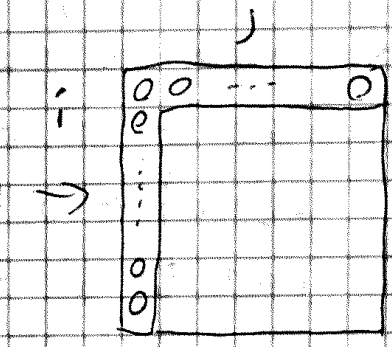
Le frecce seguono il movimento di scansione degli indici.

informazione addizionale per ricostruire la sequenza

$$b(i,j) = \begin{cases} \nwarrow & x_i = y_j \\ \leftarrow & x_i \neq y_j \text{ e } \max = LCS(i,j-1) \\ \uparrow & x_i \neq y_j \text{ e } \max = LCS(i-1,j) \end{cases}$$

$i,j > 0$

```
LCS (X, Y)
    m ← length(X)
    n ← length(Y)
    for i ← 0 to m do
        L[i,0] ← 0
    for j ← 1 to n do
        L[0,j] ← 0
    for i ← 1 to m do
        for j ← 1 to n do
            if (x_i = y_j) then
                L[i,j] ← L[i-1,j-1] + 1
                B[i,j] ← ↖
            else if L[i-1,j] ≥ L[i,j-1] then
                L[i,j] ← L[i-1,j]
                B[i,j] ← ↑
            else L[i,j] ← L[i,j-1]
                B[i,j] ← ←
    return L[m,n], B
```



) scansione row major

Complessità:

inizializzazione: O

$m \cdot n$ iterazioni, 1 solo confronto

$\rightarrow T(m,n) = \Theta(m \cdot n)$ $\rightarrow$ al più <u>quadratica</u>

(* esempio *)

Ora scriviamo il codice memoizzato (e vedremo che in alcuni casi è più efficiente)