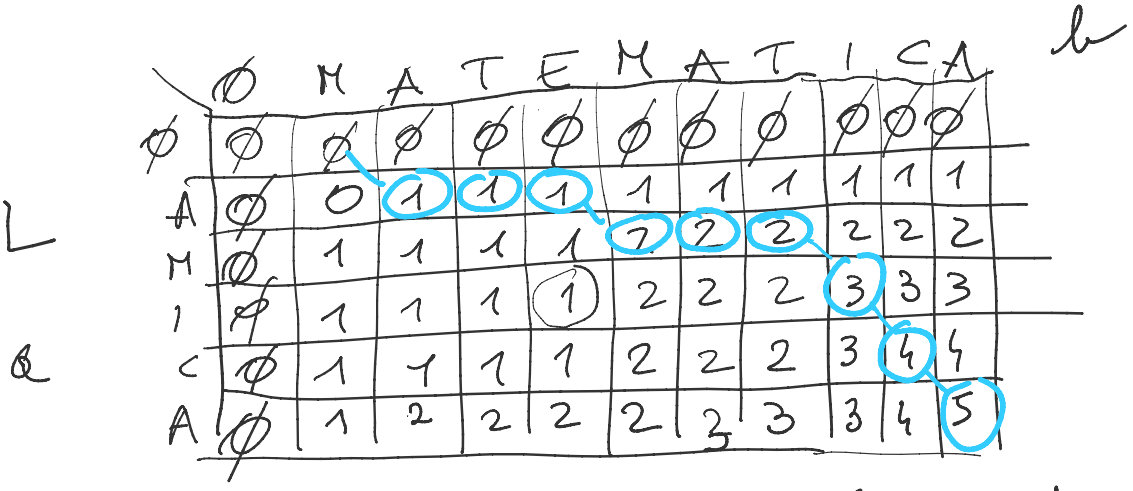
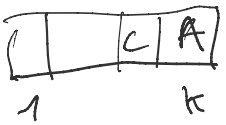
# Lezione 27 aprile

Monday, April 27, 2020 11:01 AM

LCS




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

$\Theta(n \cdot m)$ tempo

$\Theta(n \cdot m)$ spazio $\Theta(n)$ se voglio solo LCS

Per ricostruire la sequenza: percorso a ritroso. La sequenza di output in W.

```
W = new array di dim L(m,n);
i = m; j = n; k = L[m,n];
while (i > 0 && j > 0) {

if (a[i] == b[j]) {
    W[k] = a[i];
```

```
    w[k] = a[i];
    k--;
    i--;
    j--;
  } else if (L[i,j] = L[i-1,j]) i--;
      else j--;
} print(w);
```

EDIT DISTANCE

$X = x_1, \dots, x_m$

$Y = y_1, \dots, y_n$ $\quad m \cong n$

ALLINEAMENTO OTTIMO

È QUELLO A DISTANZA MINIMA CHE TOTALIZZA IL MINOR NUMERO DI ERRORI

1) Carattere errato : MISMATCH 1

2) Carattere in più in X : INSERZ. 1

3) " in più in Y : CANCELL- 1

X : ALBERO

Y : LABBRO $\quad n = m = 6$

ALLINEAMENTI OTTIMI :

ALBERO

```
A L B E R O
L A B B R O
x x = x = =
1 1   1
```

DISTANZA = 3

```
X: ␣ A L B E R O
Y: L A B B ␣ R O
x = x = x = =
```

DISTANZA = 3

```
A L ␣ B E R O
␣ L A B B R O
x = x = x = =
```

DISTANZA = 3

Principio di ottimalità - Da dimostrare

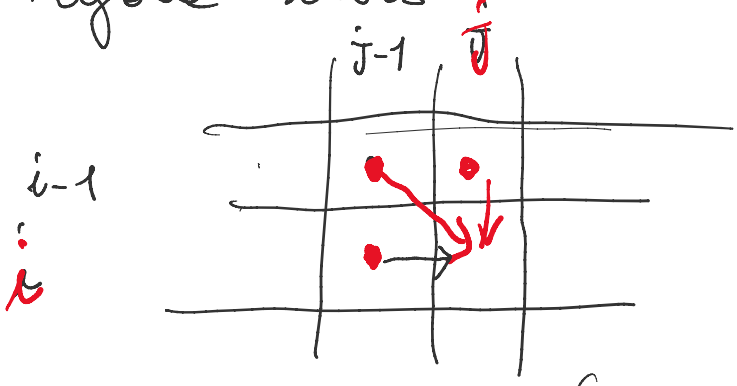
1) $E(i,j)$ allineamento ottimo di $X_1 \dots X_i$ e $Y_1 \dots Y_j$

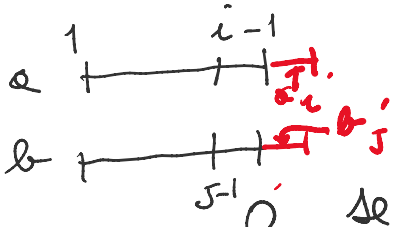
2) $E(m,n)$ $X_1 \dots X_m$ e $Y_1 \dots Y_n$

sequenze degeneri

$E(0,0) = \emptyset$ $E(i,0) = i$ $E(0,j) = j$

3) regola ricorsiva

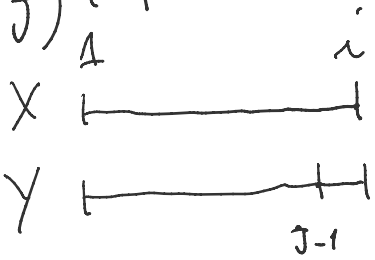



$$p_{ij} = \begin{cases} 0 & \text{se } a_i = b_j \\ 1 & \text{se } a_i \neq b_j \end{cases}$$

$$E(i,j) = \min \begin{cases} E(i-1, j-1) + p_{ij} \\ E(i, j-1) + 1 \\ E(i-1, j) + 1 \end{cases}$$

4) Tabella E
di dim $(m+1) \times (n+1)$




~~LI L~~
~~A LI~~

| | ϕ | L | A | B | B | R | O |
|---|---|---|---|---|---|---|---|
| ϕ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| A | 1 | 1 | 1 | 2 | 3 | 4 | 5 |
| L | 2 | 1 | 2 | 2 | 3 | 4 | 5 |
| B | 3 | 2 | 2 | 2 | 2 | 3 | 4 |
| E | 4 | 3 | 3 | 3 | 3 | 3 | 4 |
| R | 5 | 4 | 4 | 4 | 4 | 3 | 4 |
| O | 6 | 5 | 5 | 5 | 5 | 4 | 3 |

$p_{11} = 1$ A L

ALBERO
LABBRO = 3

LI A L B E R O
L A B B U R O
= 3

DISTANZA MINIMA = 3

$\Theta(n \cdot m)$ tempo
$\Theta(m)$ spazio per la distanza

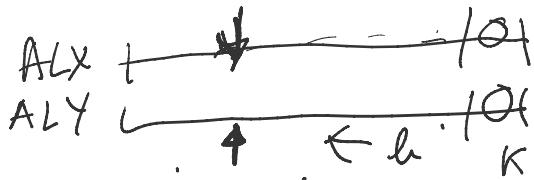
$\Theta(n\,m)$ spazio per ricostruire la sol.

Algoritmo che ricostruisce la soluzione sceglie una delle sue possibile

$k = n + m - 1$

ALX : ALLIBERO } output
ALY : -LA BBRO }

ALLINEA $(X, Y, E)$




$k = n + m - 1$ ; $h = k$ ; $i = n$ ; $j = m$ ;

while $(i > 0 \;||\; j > 0)$ { if $(i > 0 \;\&\&\; j > 0)$

match $\&\&\; [(E[i,j] == E[i-1, j-1] \;\&\&\; X[i] == Y[j]) \;||$

mismatch $(E(i,j) == E[i-1 \;\; j-1] + 1 \;\&\&\; X[i] != Y[j])$

{ ALX[h] = X[i] ; ALY[h] = Y[j] ;

i = i - 1 ; j = j - 1 ;

}

else if $(j > 0) \;\&\&\; (E[i,j] == E[i, j-1] + 1)$

{ ALX[h] = ↦ ; ALY[h] = Y[j] ;

j = j - 1 ; }

else if $(i > 0)$ {

ALX[h] = X[i] ; ALY[h] = ↦ ;

i = i - 1 ;

}

```
    h = h - 1;
  }
return h;
}
```