

STRINIGHE

PROBLEMA \rightarrow DISTANZA DI EDITING TRA 2 STRINGHE



$$x = 1001001$$

$$y = \underline{0}00\underline{0}111$$

In ERRORI RISPETTO ALLA PRIMA STRINGA
↓
DISTANZA DI HAMMING

↓
SOMMA DELLE METRICHÉ

↓
OGLI USANO LA DISTANZA DI EDITING
(o distanza di Levenshtein?)

In una stringa facciamo:

1) inserire un carattere CASA \rightarrow CASTA errore pari a 1

2) cancellazione di un carattere CASTA \rightarrow CASA errore pari a 1

3) sostituzione di un carattere CASTA \rightarrow VASTA errore pari a 1

esiste anche lo swap, l'insertione ecc...

$X \xrightarrow{\quad} y$

numero di
operazioni di editing

Come posso da $x \circ y$?

o la cosa o chiare

CASA \rightarrow CHASA \rightarrow CHESA \rightarrow CHIESA \rightarrow CHIESE

 ↑ ↑ ↑ ↑
 inserimento sostituzione inserimento sostituzione

posso anche cancellare volendo

CASA \rightarrow CAS \rightarrow CA \rightarrow C \rightarrow CH \rightarrow CHI \rightarrow CHIE \rightarrow CHIES \rightarrow
CHIESE

Se la stringa è piccola posso anche trascurare l'efficienza ma
se la stringa è gigante devo ottimizzare

$$x_i = x_1 x_2 x_3 \dots x_i = x[1 \dots i]$$

trasformiamo x in y

$$y_j = y_1 y_2 y_3 \dots y_j = y[1 \dots j]$$

$$|x| = m$$

$$|y| = m$$

Dobbiamo che

$$x_i = x_{i-1} x_i$$

$$y_j = y_{j-1} y_j$$

$$ED(x_i, y_j)$$

qualche esempio

$S^*_{i,j}$
soluzione
ottima

calcola la distanza di ED

$$x_i = y_j \rightarrow ED(x_{i-1}, y_{j-1})$$

e lo stesso
distanza ED

$$S_{i-1, j-1}$$

perché ho solo
diminuito entrambe
le stringhe

$$1) x_i = y_j \rightarrow x_{i-1} y_{j-1}$$

l'ultimo carattere di x e di y è ugualmente grande il problema
in risulta di dimensione

$$2) x_i \neq y_j$$

dove decidere quale delle 3 operazioni fare

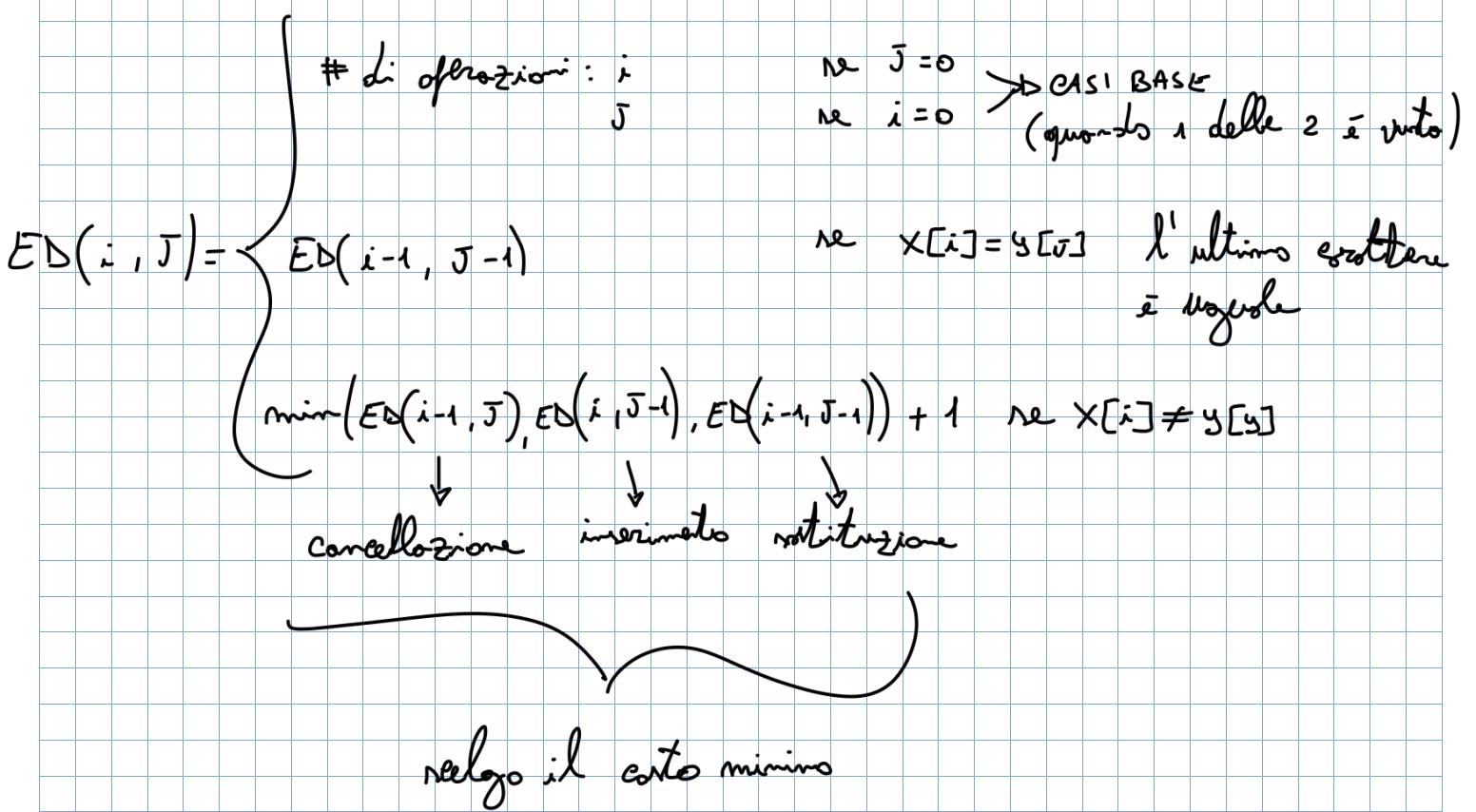
$$x_i \neq y_j \rightarrow ED(x_{i-1}, y_{j-1}) + \text{SOSTITUZIONE DI } x_i \text{ CON } y_j$$

$$x_i \neq y_j \rightarrow ED(x_{i-1}, y_j) + \text{CANCELLAZIONE DI } x_i$$

$$x_i \neq y_j \rightarrow ED(x_i, y_{j-1}) + \text{INSERIMENTO DI } y_j \text{ ALLA FINE DI } x$$

- 1) inserire un carattere
- 2) cancellazione di un carattere
- 3) sostituzione di un carattere

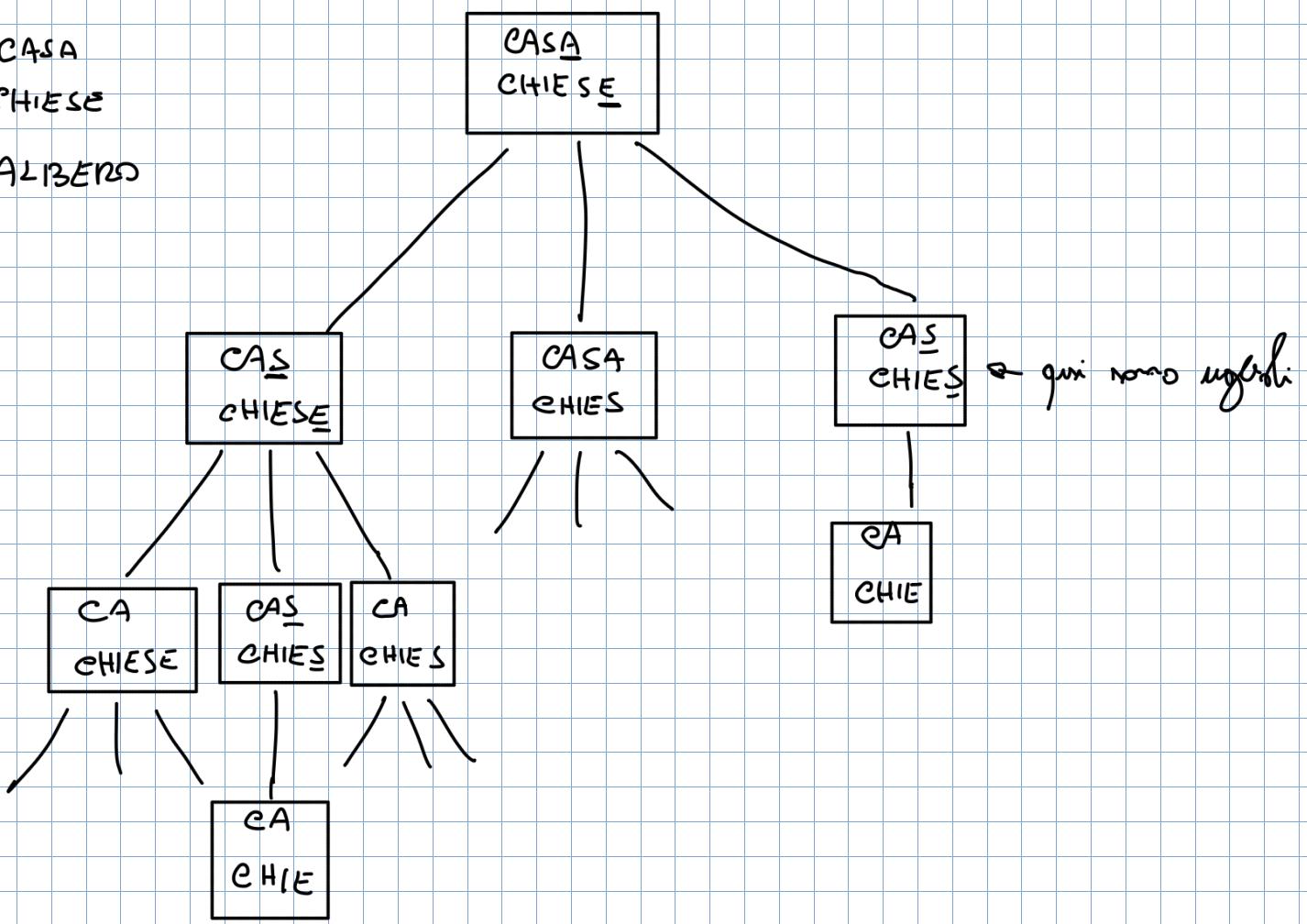
(se non ricordi forse
un esempio con cosa
e chiare)



Oraiamo la funzione ricorsiva (puro)

CASA
CHIESE

ALBERO



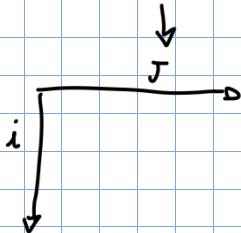
Oraiamo 2 ottoprobemi uguali, in questo caso un approccio puro, top-down

è inefficiente

ordino la soluzione quindi ricordo la fogg dinamica

	C	A	S	A	
0	0	1	2	3	4
1	1	0	1	2	3
2	2	1	1	2	3
3	3	2	2	2	3
4	4	3	3	3	3
5	5	4	4	3	4
6	6	5	5	4	4

provo la matrice in questo ordine



così non ti capisce come ha fatto questa matrice
ma se ne leggi lo pseudo codice riuscirà a ricontrollare

COSTO DI $\text{EDT}(x, y, m, n) \rightarrow O(m \times m) \rightarrow \text{TEMPO}$

$O(m \times m) \rightarrow \text{SPAZIO IN MEMORIA}$

(se volerai ottimizzare potrai anche tenere i valori nella matrice ma se solo 2 ormai e ogni volta butta i valori che non mi servono)

POINT-EDT(x, y, m, n)

$i = m$

$j = n$

WHILE ($ED[i, j] > 0$) DO

IF ($x[i] = y[j]$) THEN

$i = i - 1$

$j = j - 1$

ELSE

$$E = \min(ED[i, j-1], ED[i-1, j], ED[i-1, j-1])$$

IF $M = ED[i, j-1]$ THEN $j = j - 1$ ← STAMPIAMO: INS($y[j]$)

ELSE IF $M = ED[i-1, j]$ THEN $i = i - 1$ ← STAMPIAMO: CANCE($x[i]$)

ELSE

$i = i - 1$

$j = j - 1$

← STAMPIAMO: SOST($x[i], y[j]$)

PROBLEMA DI OTTIMIZZAZIONE

Trovare la sottosequenza più lunga di tutte

$x = A T G C E A T$

$y = G G C E A A$

longest common subsequence
otherwise ruffles

$$x[i] = y[j] \rightarrow LCS(x_{i-1}, y_{j-1}) + 1$$

$$x[i] \neq y[j] \rightarrow \emptyset$$



definizione ricorsiva del più lungo suffisso in comune

$$LCS(i, j) = \begin{cases} 0 & \text{se } i=0 \vee j=0 \\ 0 & \text{se } x[i] \neq y[j] \\ LCS(i-1, j-1) + 1 & \text{se } x[i] = y[j] \end{cases}$$

	6	6	C	C	A	T	
0	0	1	2	3	4	5	
A	1	0	0	0	0	1	
T	2	0	0	0	0	0	
G	3	0	1	1	0	0	
C	4	0	0	0	2	1	
C	5	0	0	0	1	3	
A	6	0	0	0	0	4	1
T	7	0	0	0	0	0	0

se ho lettera comune metto 1 altrimenti 0

(in qualche modo devo fare una somma e conservare il massimo temporaneo così alla fine trovo il MAX globale e quel massimo è la lunghezza del suffisso più lungo

LCS_{i-1}

LCS_i

scrivo le procedure ottimizzate usando 2 array anziché la matrice



↓

LCS(x, y, m, m)

$LCS_{i-1} = \text{new_array}(m)$

$LCS_i = \text{new_array}(m)$

FOR $J=0$ TO m DO $LCS_{i-1}[J]=0$

| FOR $i=1$ TO m DO

| $LCS_i[0]=0$

| FOR $J=1$ TO m DO

| $LCS_i[J] = \emptyset$

| IF $x[i] = y[j]$ THEN

| $LCS_i[J] = LCS_{i-1}[J-1]+1$

| IF $m < LCS_i[J]$ THEN $m = LCS_i[J]$

$LCS_{i-1} = LCS_i$

TENGO TRACCIA NEL MASSIMO
OGGI

ADESSO LCS \rightarrow longest common subsequence

$X = \underline{A} \underline{e} \underline{G} \underline{A} \underline{A} \underline{T}$

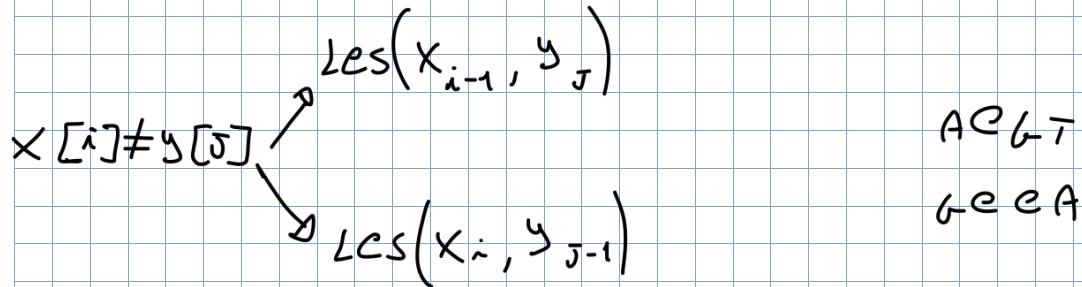
$Y = c \underline{c} \underline{A} \underline{T} \underline{A} \underline{b}$

sono delle sotto stringhe
che hanno altri caratteri
nel mezzo

↓

$$x[i] = y[j] \rightarrow \text{LCS}(x_{i-1}, y_{j-1}) + 1$$

A C G A
G C C A



$$\text{LCS}(i, j) = \begin{cases} 0 & \text{if } i=0 \vee j=0 \\ \text{LCS}(x_{i-1}, y_{j-1}) + 1 & \text{if } x[i] = y[j] \\ \max(\text{LCS}[i-1, j], \text{LCS}[i, j-1]) & \text{otherwise} \end{cases}$$

		y						
		-	C	C	A	T	A	G
		0	1	2	3	4	5	6
-	0	0	0	0	0	0	0	0
A	1	0	0	0	1	1	1	1
C	2	0	1	1	1	1	1	1
T	3	0	1	1	1	2	2	2
A	4	0	1	1	2	2	3	3
A	5	0	1	1	2	2	3	3
A	6	0	1	1	2	2	3	3

$x = \underline{A} C \underline{T} A A A$

$y = C C \underline{A T A} G$

A T A

la sotto sequenza più lunga è
lunga 3

continuano le procedure che ricontrollano le retroseguenze

PRINT-LES(L, i, j)

IF $L[i, j] = 0$ THEN

RETURN

IF $X[i] = Y[j]$ THEN

PRINT($X[i]$)

PRINT-LES($L, i-1, j-1$)

con fermo stampa le stringhe di controllo
perché non si riavvise

versione corretta

PRINT-LES(L, i, j)

IF $L[i, j] = 0$ THEN

RETURN

IF $X[i] = Y[j]$ THEN

PRINT-LES($L, i-1, j-1$)

PRINT($X[i]$)

ELSE

IF $L[i, j] = L[i-1, j]$ THEN

PRINT-LES($i-1, j$)

ELSE

PRINT-LES($i, j-1$)