# 6.6 Sequence Alignment: Ripasso Esercizi

# Edit Distance

Edit distance. [Levenshtein 1966, Needleman-Wunsch 1970]

Gap penalty $\delta$; mismatch penalty $\alpha_{pq}$ (you may assume $\alpha_{pp}=0$).

Cost = sum of gap and mismatch penalties.

Esempio:

$\delta = 3$

$\alpha_{AT}=\alpha_{CG}=3$; gli altri mismatch penalità =1 (tranne $\alpha_{pp}=0$).

| C | T | G | A | C | C | T | A | C | C | T |
|---|---|---|---|---|---|---|---|---|---|---|
| C | C | T | G | A | C | T | A | C | A | T |

$\alpha_{TC} + \alpha_{GT} + \alpha_{AG} + 2\alpha_{CA} = 5$

| - | C | T | G | A | C | C | T | A | C | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C | C | T | G | A | C | - | T | A | C | A | T |

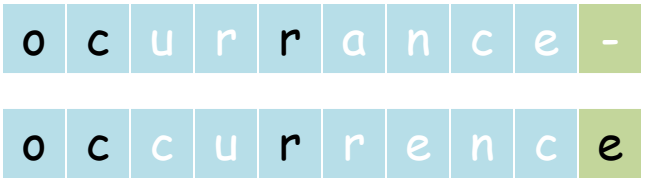
$2\delta + \alpha_{CA} = 7$

# String Similarity: example……

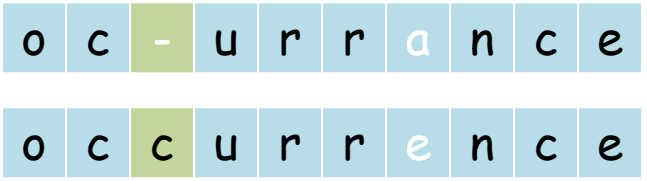
How similar are two strings?

**ocurrance**

**occurrence**




6 mismatches, 1 gap

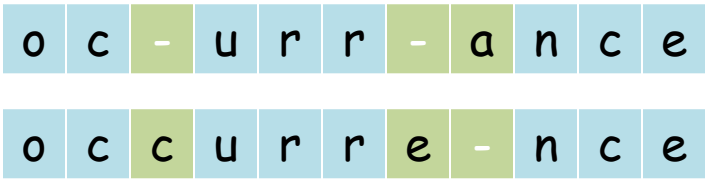



1 mismatch, 1 gap

Esempio:

$\delta = 2$

costo mismatch fra vocali differenti=1
costo mismatch fra consonanti differenti=1
costo mismatch fra vocale e consonante=3




0 mismatches, 3 gaps

# Sequence Alignment

Goal: Given two strings $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ find alignment of minimum cost.

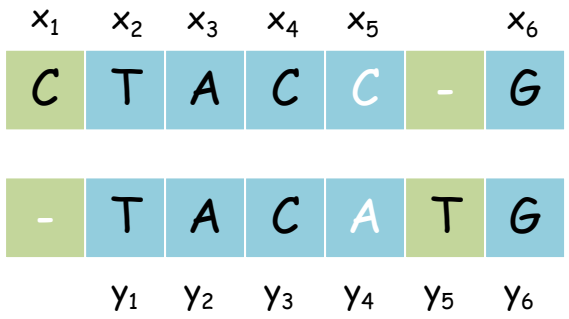
Def. An alignment M is a set of ordered pairs $x_i$-$y_j$ such that each item occurs in at most one pair and no crossings.

Def. The pair $x_i$-$y_j$ and $x_{i'}$-$y_{j'}$ cross if $i < i'$, but $j > j'$.

$$\text{cost}(M) = \underbrace{\sum_{(x_i, y_j) \in M} \alpha_{x_i y_j}}_{\text{mismatch}} + \underbrace{\sum_{i : x_i \text{ unmatched}} \delta + \sum_{j : y_j \text{ unmatched}} \delta}_{\text{gap}}$$

Ex: CTACCG vs TACATG.

Sol: $M = x_2$-$y_1$, $x_3$-$y_2$, $x_4$-$y_3$, $x_5$-$y_4$, $x_6$-$y_6$.



## Sotto-problemi

Il problema di allineare due stringhe di lunghezza m ed n si riconduce al problema di allineare due stringhe, di cui almeno una è più corta.

In generale, il sotto-problema che ci troveremo a dover risolvere è quello di allineare due prefissi delle stringhe di partenza, di lunghezza qualsiasi:

$$x_1\, x_2 \ldots x_i$$

$$y_1\, y_2 \ldots . y_j$$

Il generico sotto-problema sarà definito dagli indici i e j.

I casi base si avranno quando una (almeno) delle due stringhe è vuota; in tal caso per i simboli dell'altra stringa si potranno avere solo gap

## Soluzione ottimale OPT

Consideriamo un allineamento ottimale OPT delle stringhe

$$x_1\, x_2 \ldots x_i \quad \text{e} \quad y_1\, y_2 \ldots y_j$$

Guardiamo l'allineamento in OPT degli ultimi caratteri $x_i$ e $y_j$.

Sono possibili 2 casi: $x_i$ e $y_j$ sono in corrispondenza oppure no; nel secondo caso almeno uno dei 2 è in corrispondenza di uno spazio vuoto (altrimenti crossing).

CASO 1: $x_i$ e $y_j$ sono in corrispondenza: $x_1\, x_2 \ldots x_{i-1}\ x_i$

$y_1\, y_2 \ldots y_{j-1}\ y_j$

Caso 2a: $x_i$ non corrisponde a nessun carattere:

$x_1\, x_2 \ldots x_{i-1}\ x_i$

$y_1\, y_2 \ldots y_j\ -$

Caso 2b: $y_j$ non corrisponde a nessun carattere:

$x_1\, x_2 \ldots x_i\ -$

$y_1\, y_2 \ldots y_{j-1}\ y_j$

# Sequence Alignment: Problem Structure

Def. OPT(i, j) = min cost of aligning strings $x_1\ x_2\ \ldots\ x_i$ and $y_1\ y_2\ \ldots\ y_j$.

Case 1: OPT matches $x_i$-$y_j$.

pay mismatch for $x_i$-$y_j$ + min cost of aligning two strings $x_1\ x_2\ \ldots\ x_{i-1}$ and $y_1\ y_2\ \ldots\ y_{j-1}$

Case 2a: OPT leaves $x_i$ unmatched.

pay gap for $x_i$ and min cost of aligning $x_1\ x_2\ \ldots\ x_{i-1}$ and $y_1\ y_2\ \ldots\ y_j$

Case 2b: OPT leaves $y_j$ unmatched.

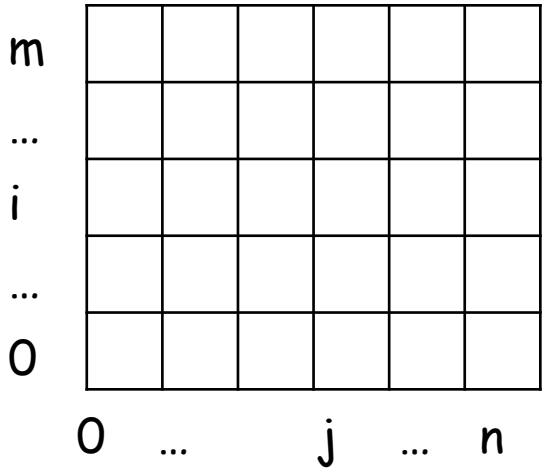
pay gap for $y_j$ and min cost of aligning $x_1\ x_2\ \ldots\ x_i$ and $y_1\ y_2\ \ldots\ y_{j-1}$

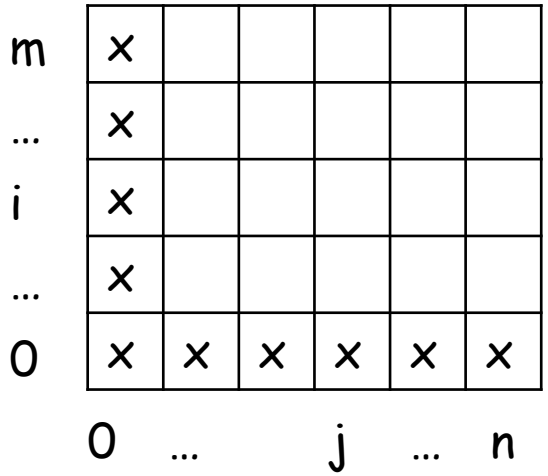
$$OPT(i,j) = \begin{cases} j\delta & \text{if } i = 0 \\ \min \begin{cases} \alpha_{x_i y_j} + OPT(i-1, j-1) \\ \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \end{cases} & \text{otherwise} \\ i\delta & \text{if } j = 0 \end{cases}$$

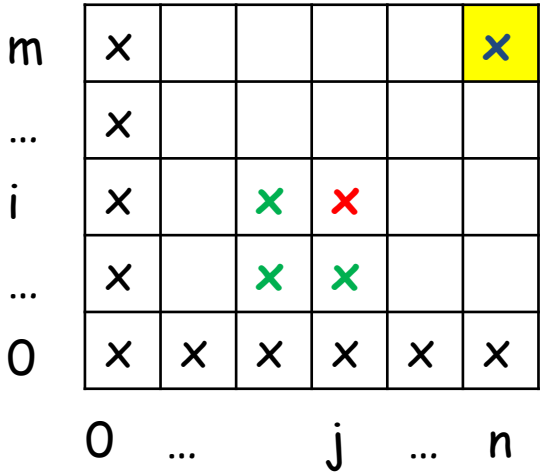
# Implementazione con algoritmo iterativo

Uso una tabella M per contenere OPT(i,j) per i=0,1,…m, j=0,1,…,n.

Comincio con l'inserire i valori dei casi base, i=0 e j=0, una riga e una colonna.




Scorro tutte le altre celle della tabella in modo che quando devo riempire la cella M[i,j], i valori necessari al calcolo siano già inseriti nelle corrispondenti celle. I valori necessari sono M[i-1,j-1], M[i,j-1] e M[i-1,j]. Posso scorrere per righe da 1 a m e per colonne da 1 a n.

La soluzione al problema di allineare X e Y sarà in M[m,n], che restituisco alla fine.

# Sequence Alignment: Algorithm

```
Sequence-Alignment(m, n, x1x2...xm, y1y2...yn, δ, α) {
   for j = 0 to n
      M[0,j] = jδ
   for i = 0 to m
      M[i,0] = iδ
   for i = 1 to m
      for j = 1 to n
         M[i,j] = min(α[xi,yj] + M[i-1,j-1],
                      δ + M[i-1,j],
                      δ + M[i,j-1])
   return M[m, n]
}
```

Analysis. $\Theta(mn)$ time and space.

# Esempio

$$OPT(i,j) = \begin{cases} j\delta & \text{if } i = 0 \\ \min \begin{cases} \alpha_{x_i y_j} + OPT(i-1, j-1) \\ \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \end{cases} & \text{otherwise} \\ i\delta & \text{if } j = 0 \end{cases}$$

Esempio: X = mean, Y= name

$\delta$ = 2

costo mismatch fra vocali differenti=1

costo mismatch fra consonanti differenti=1

costo mismatch fra vocale e consonante=3

M[0,0]=0, M[0,1]=2, M[0,2]=4, .....

M[1,0]=2, M[2,0]=4, .....

M[1,1]=min{$\alpha_{m\,n}$ + M[0,0], $\delta$ + M[0,1], $\delta$+M[1,0]}=
= min { **1+0**, 2+2, 2+2} = **1**

M[1,2]=min{$\alpha_{m\,a}$ + M[0,1], $\delta$ + M[0,2], $\delta$+M[1,1]}=
= min { 3+ 2, 2+4, **2+1**} = **3**

M[1,3]=min{$\alpha_{m\,m}$ + M[0,2], $\delta$ + M[0,3], $\delta$+M[1,2]}=
= min { **0+4**, 2+6, 2+3} = **4**

.....

M[4,4]= min{$\alpha_{n\,e}$ + M[3,3], $\delta$ + M[3,4], $\delta$+M[4,3]}
= min { 3+ 5, 2+5, **2+4**} = **6**

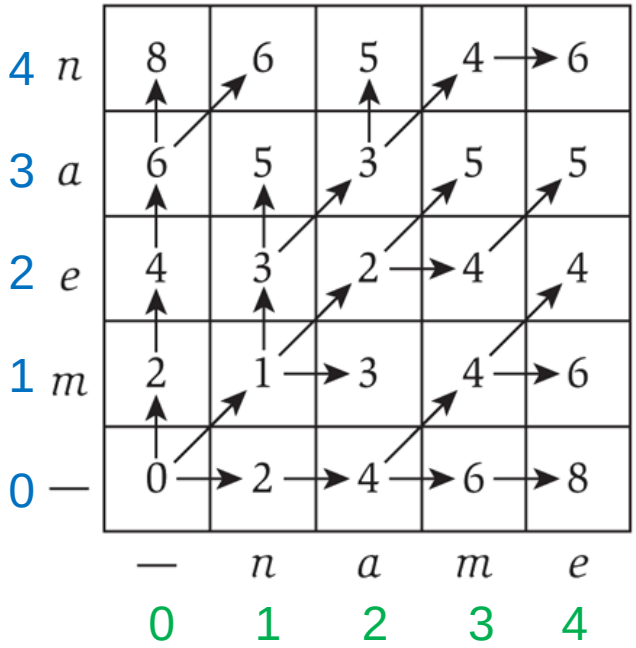



**Figure 6.18** The OPT values for the problem of aligning the words *mean* to *name*.

# Ricostruzione dell'allineamento

$$OPT(i,j) = \begin{cases} j\delta & \text{if } i = 0 \\ \min \begin{cases} \alpha_{x_i y_j} + OPT(i-1, j-1) \\ \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \end{cases} & \text{otherwise} \\ i\delta & \text{if } j = 0 \end{cases}$$

Esempio: X = mean, Y= name

$\delta$ = 2

costo mismatch fra vocali differenti=1

costo mismatch fra consonanti differenti=1

costo mismatch fra vocale e consonante=3

La freccia nella casella (i,j) proviene dalla casella usata per ottenere il minimo

M[4,4] =

= min{$\alpha_{n\,e}$+M[3,3], $\delta$ + M[3,4], $\delta$+M[4,3] } =

= min { 3+ 5, 2+5, 2+4} = 6

Inserisco freccia da M[4,3] → M[4,4]

Per ricostruire l'allineamento seguiamo il percorso all'indietro nella matrice

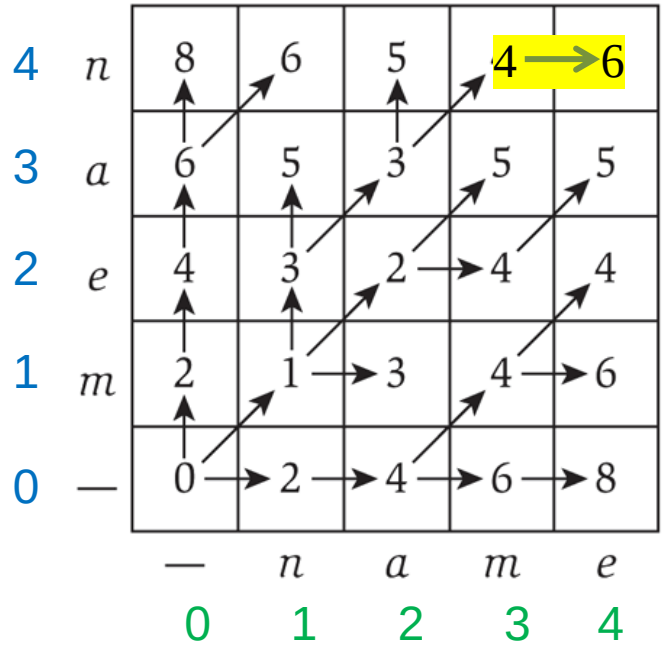



**Figure 6.18** The OPT values for the problem of aligning the words *mean* to *name*.

# Ricostruzione soluzione ottimale

Per ricostruire un allineamento ottimale seguiamo il percorso all'indietro nella matrice

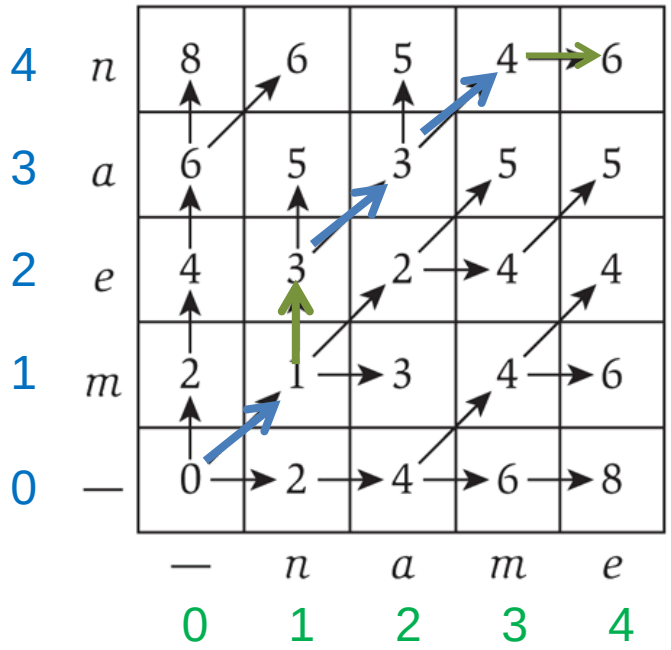
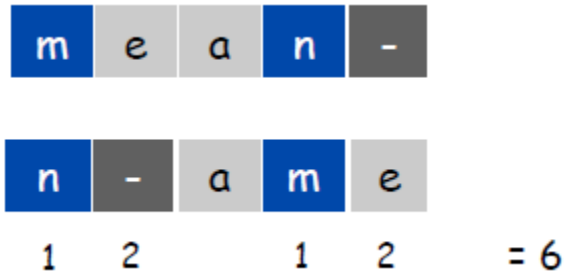



**Figure 6.18** The OPT values for the problem of aligning the words *mean* to *name*.




M[4,4]= min{$\alpha_{n\,e}$ + M[3,3], $\delta$ + M[3,4], $\delta$+M[4,3]}=
= min { 3+ 5, 2+5, **2+4**} = **6**

## Una visione grafica del problema del sequence alignment

Ad $X = x_1\ x_2\ x_3$ e $Y = y_1\ y_2\ y_3\ y_4$ associamo il grafo $G_{XY}$ seguente:

nodo (i,j) in corrispondenza di $x_i$ e $y_j$

costi archi orizzontali e verticali = $\delta$

costo arco diagonale verso (i,j) = $\alpha_{xi,\ yj}$

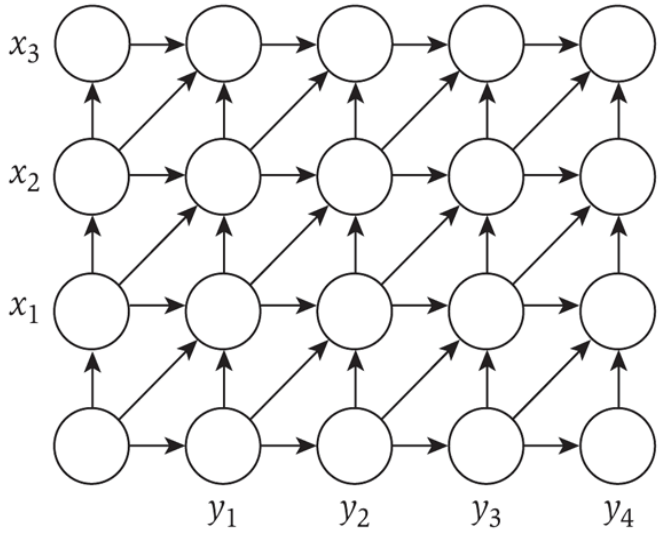



**Figure 6.17** A graph-based picture of sequence alignment.

# Un esempio

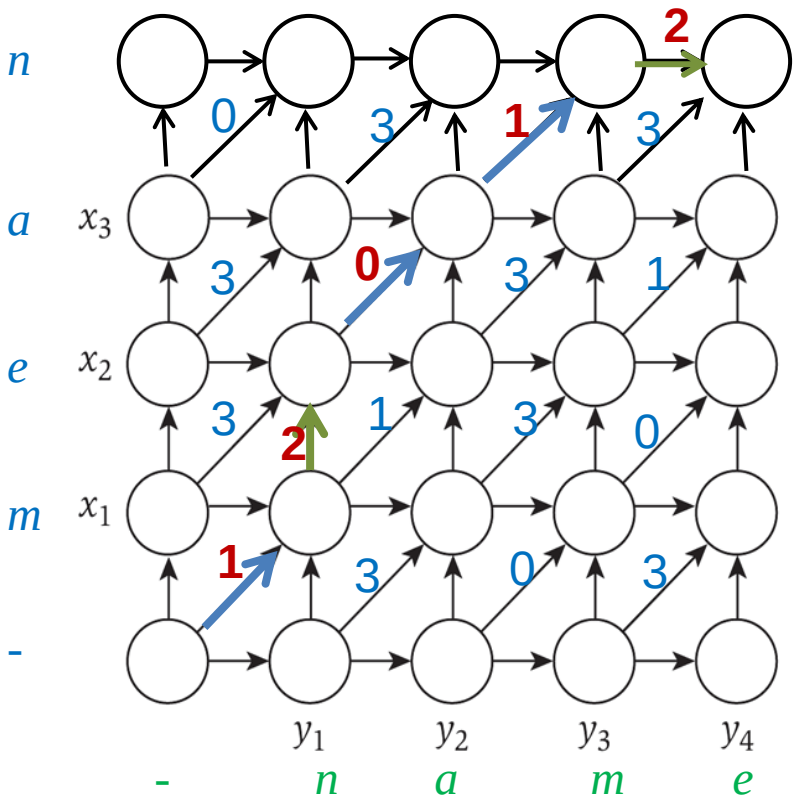



**Figure 6.17** A graph-based picture of sequence alignment.

costi archi orizzontali e verticali = $\delta$
costo arco diagonale verso (i,j) = $\alpha_{xi, yj}$

$\delta = 2$
costo mismatch fra vocali differenti=1
costo mismatch fra consonanti differenti=1
costo mismatch fra vocale e consonante=3

Un allineamento di costo minimo fra X e Y può essere ottenuto da un cammino di costo minimo in $G_{XY}$ dal nodo in basso a sinistra a quello in alto a destra.
Prova per induzione (fare)

Studieremo algoritmi per la ricerca di cammini di costo minimo in un grafo, ma non apporteranno miglioramenti al tempo di esecuzione.

## Esercizio (da slide precedente)

Provare per induzione che:

Un allineamento di costo minimo fra X e Y può essere ottenuto da un cammino di costo minimo in $G_{XY}$ dal nodo in basso a sinistra a quello in alto a destra.

# Sequence Alignment: Linear Space

Analysis. $\Theta(mn)$ time and space.

English words or sentences: $m, n \leq 10$.

Computational biology: $m = n = 100{,}000$. Then, 10 billions ops OK, but 10GB array?

Q. Can we avoid using quadratic space?

Easy. Optimal value in $O(m + n)$ space and $O(mn)$ time.

Compute $OPT(i, \cdot)$ from $OPT(i-1, \cdot)$.

No longer a simple way to recover alignment itself.

Theorem. [Hirschberg 1975]

Optimal alignment in $O(m + n)$ space and $O(mn)$ time.

Clever combination of divide-and-conquer and dynamic programming.

Inspired by idea of Savitch from complexity theory.

## Programmazione dinamica (pseudocodice) (svolto)

Si supponga che la soluzione ad un certo problema (a noi ignoto) sia data, per un certo intero n positivo, dal massimo fra i valori OPT(n,R) e OPT(n,B) definiti ricorsivamente come segue (R sta per Rosso e B sta per Blu):

$$OPT(1, R) = 2$$

$$OPT(1, B) = 1$$

$$OPT(i, R) = OPT(i - 1, B) + 1, \text{ se } i > 1$$

$$OPT(i, B) = \max \{OPT(i, R) - 1, OPT(i - 1, R)\}, \text{ se } i > 1$$

a) Calcolare i valori di OPT(i, R) e OPT(i, B) per ogni i=1, 2, …, 5, organizzandoli in una **tabella**.

b) Scrivere lo pseudocodice di un algoritmo **ricorsivo** per il calcolo della soluzione al problema.

c) Scrivere lo pseudocodice di un algoritmo di **programmazione dinamica** per il calcolo della soluzione al problema. Analizzarne la complessità di **tempo** e di **spazio**, giustificando la risposta.

# Appello 22 febbraio 2016

3)

Si considerino le stringhe

X= $x_1\, x_2\, x_3\, x_4\, x_5$ = mamma e Y= $y_1\, y_2\, y_3$ = mia.

Se il costo di un *gap* è 5, il costo di un *mismatch* fra due vocali differenti è 3, fra due consonanti differenti è 4 e fra vocale e consonante è 7, allora il costo dell'allineamento $x_1$–$y_1$ , $x_3$–$y_2$, $x_5$–$y_3$ è:

A. 7

B. 14

C. 17

D. Nessuna delle risposte precedenti

# Esercizio

Si considerino le stringhe
X= $x_1\, x_2\, x_3\, x_4\, x_5$ = mamma e Y= $y_1\, y_2\, y_3$ = mia.
Si supponga che il costo di un *gap* sia 5, il costo di un *mismatch* fra due vocali differenti sia 3, fra due consonanti differenti è 4 e fra vocale e consonante è 7.

Calcolare il costo minimo di un allineamento delle stringhe X e Y, applicando l'algoritmo studiato.

# Appello 24 gennaio 2017

**Quesito 1** (24 punti)

Si consideri la seguente funzione c(i,j) definita per $1 \le i, j \le n$ da:

$c(1,j)=1$ per ogni $1 \le j \le n$, $\quad c(i,1)=3$ per ogni $2 \le i \le n$,

$c(2,j)=2$ per ogni $2 \le j \le n$, $\quad c(i,2)=4$ per ogni $3 \le i \le n$,

$c(i,j)= \max \{ 3 \times c(i-2, j), c(i, j-1) - 2 \}$ per ogni $3 \le i, j \le n$.

a) Disegnare la matrice c per n=4

b) Scrivere lo pseudocodice di un algoritmo ricorsivo per il calcolo di c(n,n) e indicarne la complessità di tempo (non è necessaria una analisi dettagliata).

c) Scrivere lo pseudocodice di un algoritmo di programmazione dinamica per il calcolo di c(n,n) ed analizzarne la complessità. E' necessario giustificare la risposta.

# Prima prova intercorso 2017/18

**Quesito 3** (16 punti)

Si consideri il problema dello *scheduling* di intervalli pesato.

Ricostruire uno *scheduling* ottimale per il problema dato su un insieme di intervalli $S=\{1, 2, \ldots, 6\}$ ordinato secondo il tempo di fine crescente degli intervalli (cioè $f_1 \leq f_2 \leq \ldots \leq f_6$), sapendo che i pesi degli intervalli sono rispettivamente $w_1=12$, $w_2=2$, $w_3=8$, $w_4=9$, $w_5=3$, $w_6=10$, che i valori della funzione p sono $p(1)=0$, $p(2)=0$, $p(3)=2$, $p(4)=0$, $p(5)=1$, $p(6)=4$ e l'array M calcolato dall'algoritmo di programmazione dinamica studiato è $M[0 .. 6] = [0, 12, 12, 20, 20, 20, 30]$. E' necessario giustificare la risposta.

Si noti che non occorre conoscere i valori dei tempi di inizio e di fine degli intervalli per ricostruire la soluzione.

## Giornata di seminari (PD)

Vi state occupando di organizzare una giornata di seminari nell'Aula Magna della vostra università. Avete avuto la disponibilità di vari relatori a tenere un loro intervento; ognuno ha specificato da che ora a che ora si terrebbe il suo seminario. Purtroppo, non riuscite ad organizzare la giornata in modo da inserire tutti i relatori. Dovete perciò scegliere alcuni fra i relatori in modo che i loro seminari possano essere svolti nell'aula senza sovrapposizioni di orario. Volete inoltre fare in modo che sia massimo il **tempo** totale di utilizzo effettivo dell'aula.

a) Definire il problema computazionale, specificandone i dati in ingresso e in uscita.

b) Indicare se si tratta di un problema studiato e, se sì, quale.

c) Progettare un algoritmo di programmazione dinamica che risolve il problema e valutarne la complessità. Si potrà ottenere il massimo della votazione solo se l'algoritmo è descritto tramite pseudo-codice ed è discussa la sua correttezza.