Fabrizio Luccio. Informatica per le Biotecnologie

7 CONFRONTO TRA SEQUENZE

Il paradigma della programmazione dinamica introdotto nella dispensa 6 è alla base della risoluzione di molti problemi di **confronto fra sequenze** che si incontrano nella elaborazione di testi e nella biologia molecolare. Iniziamo con lo studio del problema della *edit distance* (distanza di edizione, in una versione in italiano scarsamente usata): lo schema algoritmico potrà poi essere facilmente modificato per risolvere altri problemi sulle sequenze.

EDIT DISTANCE. Date due sequenze di caratteri X,Y trovare un *allineamento ottimo* delle due che corrisponda alla *minima distanza* tra X e Y, ove la distanza è definita secondo le regole seguenti:

• si ammette che nelle due sequenze possano essere inseriti spazi indicati con - e che, per ogni posizione nelle due sequenze, il carattere o lo spazio che appare nella X sia posto in corrispondenza (allineato) con il carattere o lo spazio nella stessa posizione della Y;

• la distanza è la somma delle distanze tra coppie di caratteri o spazi, uno in X e l'altro in Y, in posizioni corrispondenti nell'allineamento: caratteri uguali hanno distanza zero, indicata con Ø (**match**); caratteri diversi hanno distanza 1 (**mismatch**); un carattere allineato a uno spazio ha distanza 1 (**space**);

• la presenza di uno spazio si interpreta come l'avvenuta **cancellazione** di un carattere dalla sequenza, o come l'**inserzione** di un carattere nell'altra in posizione corrispondente: per questo motivo non si considerano spazi in posizioni corrispondenti di X,Y;

• la **edit distance** tra X e Y è la distanza relativa all'allineamento (o agli allineamenti) che minimizza tale distanza.

Valga come esempio la distanza tra le sequenze alfabetiche X = ALBERO e Y = LABBRO. Tre allineamenti con distanza uguale a 3 (che come vedremo è la minima possibile, cioè la edit distance) sono riportati qui sotto. Il segno • indica le coppie nell'allineamento che danno luogo a distanza locale 1; per tutte le altre coppie si verifica un match e la distanza locale è Ø.

```
A L B E R O       A L - B E R O       - A L B E R O
L A B B R O       - L A B B R O       L A B B - R O
• • •             • • • •             • • • •
```

Indicate le sequenze come $X = x_1\, x_2 \ldots x_n$, $Y = y_1\, y_2 \ldots y_m$, il calcolo impiega una matrice M di dimensioni $(n+1)\times(m+1)$ ove **M[i,j] indica la edit distance tra il prefisso $x_1\, x_2 \ldots x_i$ di X e il prefisso $y_1\, y_2 \ldots y_j$ di Y.** Dunque i caratteri di X corrispondono alle righe di M, i caratteri di Y alle colonne, e l'ultimo valore M[n,m] indica la edit distance tra le due sequenze. La riga e la colonna Ø corrispondono a prefissi vuoti, cioè X e Y non sono ancora stati esaminati, quindi la M si inizializza sulla riga Ø e la colonna Ø ponendo:

$M[Ø,j]=j$ per $Ø\le j\le m$, $M[i,Ø]=i$ per $Ø\le i\le n$,

valori che corrispondono ad affermare che il prefisso vuoto di X allineato con il prefisso $y_1\, y_2 \ldots y_j$ di Y corrisponde a una distanza j, e che il prefisso vuoto di Y allineato con il prefisso $x_1\, x_2 \ldots x_i$ di X corrisponde a una distanza i. Per le sequenze viste sopra il valore M[Ø,3] = 3 corrisponde all'allineamento del prefisso LAB di LABBRO con tre spazi --- inseriti prima di ALBERO.

Indicata con p(i,j) la distanza locale tra x_i e y_j, cioè posto **p(i,j)=Ø se $x_i=y_j$ (match), p(i,j)=1 se $x_i\neq y_j$ (mismatch)** secondo i valori indicati sopra, gli elementi M[i,j] con $1\le i\le n$ e $1\le j\le m$ si calcolano progressivamente, riga per riga, con la formula ricorsiva:

$$\mathbf{M[i,j] = \min\{M[i,j-1]+1,\ M[i-1,j]+1,\ M[i-1,j-1]+p(i,j)\}} \qquad (1)$$

che si spiega notando che l'allineamento ottimo tra i prefissi $x_1\, x_2 \ldots x_i$ e $y_1\, y_2 \ldots y_j$ si può costruire dagli allineamenti ottimi per i prefissi privati dell'ultimo carattere, confrontando $x_1\, x_2 \ldots x_i$ con $y_1\, y_2 \ldots y_{j-1}$–; $x_1\, x_2 \ldots x_{i-1}$– con $y_1\, y_2 \ldots y_j$; e $x_1\, x_2 \ldots x_i$ con $y_1\, y_2 \ldots y_j$; e scegliendo tra essi quello che genera distanza minima. Avremo per il nostro esempio la seguente matrice:

	Ø	L	A	B	B	R	O
Ø	Ø	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

L'ultimo valore M[6,6] = 3 indica la edit distance tra le due sequenze, che come già affermato è uguale a tre. Si noti che la relazione (1) è ricorsiva ma l'algoritmo di costruzione di M è iterativo e segue uno schema di programmazione dinamica. Tale algoritmo ha **complessità quadratica** $\Theta(nm)$ perché richiede di costruire una matrice M di dimensioni $(n+1)\times(m+1)$, e il valore in ciascuna cella è calcolato in tempo costante, perché i tre valori che appaiono nella formula ricorsiva indicata sopra sono stati già calcolati in passi precedenti e memorizzati in M.

La formulazione dell'algoritmo in forma di un programma che chiameremo ED, fa uso di due vettori X, Y di dimensioni n+1 e m+1 che contengono le sequenze da confrontare a partire dalla cella 1 (la cella 0 non è utilizzata), e costruisce la matrice M secondo lo schema già descritto a parole:

```
ED(X,Y)
// Calcolo della edit distance tra due sequenze contenute nei
vettori X,Y, costruendo la matrice M[0:n,0:m]

   for (i=0,i≤ n,i++) M[i,0]=i;  //inizializza la colonna 0
   for (j=0,j≤ m,j++) M[0,j]=j;  //inizializza la riga 0
   for (i=1,i≤ n,i++)
       for (j=1,j≤ m,j++)
         { if (X[i]=Y[j]) p=0; else p=1;
           M[i,j]=min(M[i-1,j]+1,M[i-1,j-1]+p,M[i,j-1]+1); }
   return M[n,m];
```

Gli allineamenti che danno luogo alla edit distance M[n,m] (M[6,6] nell'esempio) si ricostruiscono risalendo all'indietro nella matrice, dalla casella [n,m] fino alla casella [Ø,Ø]. Da M[i,j] si risale a M[i-1,j-1] se questi due valori sono uguali e $x_i=y_j$ (per esempio si risale da M[6,6] a M[5,5] entrambi = 3, che indica un match tra caratteri); oppure si risale al valore di M uguale a M[i,j]-1 tra i tre adiacenti che lo precedono: in questo caso vi possono essere più alternative, corrispondenti ad allineamenti di uguale distanza. Nell'esempio si ricostruiscano i tre allineamenti ottimi tra ALBERO e LABBRO già indicati in precedenza, partendo da M[6,6]: ci proponiamo di trovarne uno solo perché le soluzioni potrebbero essere moltissime.

Se si richiede di ricostruire un solo allineamento ottimo l'algoritmo richiede tempo $\Theta(n+m)$ per tracciare il percorso all'indietro dalla casella [n,m] alla [Ø,Ø]: infatti n+m è il numero massimo di passi che l'algoritmo può compiere sulla matrice (nel caso che questi avvengano sempre in orizzontale o in verticale) e ciascun passo richiede un numero costante di confronti.

Un programma per eseguire questo compito, che chiameremo ALLINEA, fa uso come il precedente ED dei due vettori X,Y che contengono le sequenze da confrontare e della matrice M prodotta da ED, e costruisce l'allineamento ottimo in due nuovi vettori ALX,ALY che conterranno le due sequenze inclusi gli spazi che devono apparire in tale allineamento. Per esempio, per il secondo allineamento ottimo visto sopra avremo:

```
ALX:    A L - B E R O
ALY:    - L A B B R O
```

I due vettori contengono lo stesso numero di elementi (sette, nell'esempio), ma questo numero non è noto a priori; quindi essi saranno definiti per k=n+m elementi che sono certamente sufficienti. Si noti che i due vettori vengono riempiti a partire dall'ultima casella k (in corrispondenza all'esame di M[n,m]), per posizioni decrescenti e fino a una casella h, $1 \leq h \leq k$, che conterrà l'inizio delle sequenze allineate: gli elementi nelle caselle 0...h-1 non avranno alcun significato.

```
ALLINEA(X,Y,M)
// Costruzione dell'allineamento ottimo tra due sequenze contenute
nei vettori X,Y utilizzando la matrice M costruita dal programma
ED. Il valore di h, restituito dall'algoritmo attraverso la frase
return, indica da che punto interpretare il contenuto dei vettori
ALX, ALY  //
   k=n+m-1; h=k; i=n; j=m;
   while ((i>0) or (j>0))
       {if (((i>0) and (j>0))
             and ((M[i,j]==M[i-1,j-1])&&(X[i]==Y[j])
                 or ((M[i,j]==M[i-1,j-1]+1) and (X[i]!=Y[j])))
                    {ALX[h]=X[i]; ALY[h]=Y[j]; i=i-1; j=j-1;}
        else {if ((j>0) and (M[i,j]==M[i,j-1]+1))
                 {ALX[h]= -; ALY[h]=Y[j]; j=j-1;}
              else if (i>0)  //deve essere M[i,j]=M[i-1,j]+1
                    {ALX[h]=X[i]; ALY[h]= -; i=i-1;}
             }
         h=h-1;
        }
   return h;
```

Esercizio 1. Si esamini attentamente il programma ALLINEA per comprendere come funziona e con quale criterio viene scelto l'allineamento tra i tanti possibili. Nell'esempio ALBERO-LABBRO quale allineamento sceglierà ALLINEA?

A titolo di ulteriore esempio sull'uso e sul significato della matrice M relativamente alla edit distance si consideri l'elemento M[3,6]=4 corrispondente al confronto tra ALB e LABBRO cui corrisponde l'allineamento ottimo:

```
- A L B - -
L A B B R O
•   •     • •
```

Oppure l'elemento M[4,3]=3 corrispondente al confronto tra ALBE e LAB cui corrispondono gli allineamenti ottimi:

```
A L B E       A L B E       - A L B E
- L A B       L A B -       L A - B -
•   • •       • •   •       •   •   •
```

Il calcolo della edit distance può essere immediatamente adattato a risolvere il problema con diversi valori del costo delle distanze tra singoli caratteri. Si tratta unicamente di modificare la relazione di ricorrenza (1) in funzione dei nuovi costi e inserire questi costi negli algoritmi ED e ALLINEA.

Esercizio 2. Si calcoli nuovamente la distanza tra ALBERO e LABBRO e il loro allineamento ottimo ponendo $p(i,j)=2$ per $x_i \neq y_j$ e lasciando invariati gli altri costi.

Come già detto l'algoritmo di base per il calcolo di distanza può essere applicato alla risoluzione di altri problemi di confronto tra sequenze con semplicissime trasformazioni.
Uno di questi problemi, di grande importanza sia per gli editori di testo che per la biologia molecolare, è il seguente:

RICERCA APPROSSIMATA DI UN PATTERN IN UN TESTO. Date una sequenza Y di m caratteri detta **testo** e una sequenza X di n caratteri detta **pattern**, con n << m, trovare tutte le apparizioni di X in Y ammettendo che queste apparizioni possano contenere qualche differenza e quindi la distanza tra la X e la sottosequenza di Y con cui si confronta la X possa non essere zero.

A tale scopo si può impiegare l'algoritmo di edit distance, notando però che **M[i,j] dovrà ora indicare la edit distance tra il prefisso $x_1\, x_2 \ldots x_i$ di X e le sottosequeze di Y che terminano in posizione j**, cioè le sottosequenze $y_k\, y_{k+1} \ldots y_j$ **con** $\emptyset \leq k \leq j$, ove sia i prefissi di X che le sottosequenze di Y possono contenere degli spazi e il

valore di k non è noto a priori. Per ottenere questo risultato occorre unicamente inizializzare la riga zero con tutti Ø, cioè porre:

$$M[Ø,j]=Ø, \quad \text{per } Ø \leq j \leq m$$

che corrisponde ad assegnare edit distance zero all'allineamento tra il prefisso vuoto della X e qualunque prefisso della Y, perché il pattern può apparire in qualsiasi punto del testo.

Chiariamo questi punti con un esempio. La seguente matrice è relativa alla ricerca delle apparizioni di X = RAT in Y = SERRATURA.

	Ø	S	E	R	R	A	T	U	R	A
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
R	1	1	1	Ø	Ø	1	1	1	Ø	1
A	2	2	2	1	1	Ø	1	2	1	Ø
T	3	3	3	2	2	1	Ø	1	2	1

I valori rilevanti son quelli nell'ultima riga che indicano la minima edit distance tra la X e le sottosequenze di Y che terminano in quella posizione. Tra questi si sceglieranno i più bassi, ovvero quelli con un valore inferiore a un limite prefissato. Nell'esempio il valore M[3,6]=Ø corrisponde all'apparizione esatta di X=RAT nelle posizioni 4,5,6 di Y=SERRATURA. Il valore M[3,4]=2 corrisponde al migliore allineamento tra RAT e le sottosequenze di Y che terminano in posizione 4, cioè ai tre allineamenti:

R	A	T	R	A	T	R	A	T
R	R	-	R	-	R	R	-	-
	•	•		•	•		•	•

Per determinare gli allineamenti tra la X e le sottosequenze di Y si procede come nel caso precedente, risalendo dalla posizione considerata nell'ultima riga fino alla riga Ø, ma senza dover poi recedere fino alla posizione [Ø,Ø]. Per esempio l'ultimo elemento M[3,9]=1 corrisponde all'allineamento di RAT con RA- (il calcolo di M[3,9] è stato eseguito scendendo in verticale dalla casella M[2,9]). Si noti che tra questi allineamenti alcuni potrebbero contenere spazi nella X, anche se ciò non si verifica in questo esempio.

Un problema rilevante in biologia molecolare e legato molto strettamente a quello della edit distance è noto come:

GLOBAL COMPARISON. Date due sequenze X,Y trovare un allineamento di **massima similarità**, ove la similarità è definita secondo le regole seguenti:

• come in precedenza, nelle due sequenze possano essere inseriti degli spazi e nell'allineamento non si considerano spazi in posizioni corrispondenti di X,Y;

• il peso di un allineamento è la somma dei pesi tra coppie di caratteri, o tra caratteri e spazi, in posizioni corrispondenti nell'allineamento, ove caratteri uguali (**match**) hanno peso positivo +1, caratteri diversi (**mismatch**) hanno peso negativo -1; un carattere allineato a uno spazio (**space**) ha peso negativo -1; questi pesi possono essere variati senza modificare la struttura dell'algoritmo allo scopo di assegnare rilevanza diversa ai tre tipi di differenze (mismatch, spazio in X, spazio in Y), o pesare diversamente il valore del match;

• la **similarità** tra X e Y è il peso relativo all'allineamento (o agli allineamenti) che massimizza tale peso.

Dunque in questo problema si attribuisce un credito ai match e un debito alle differenze e il valore di similarità è tanto più alto quanto più i match prevalgono sulle differenze. Mentre la edit distance tra due sequenze è funzione solo delle loro differenze, la similarità dipende in modo rilevante anche dal numero di match. Così coppie di sequenze con le stesse differenze puntuali avranno uguale edit distance ma similarità tanto più alta quanto più sono lunghe poiché coincideranno in tutti gli altri caratteri.

Come in precedenza si impiega l'algoritmo di base sulla matrice M ove M[i,j] indica ora la similarità tra il prefisso $x_1\, x_2 \ldots x_i$ di X e il prefisso $y_1\, y_2 \ldots y_j$ di Y contenenti eventualmente degli spazi, quindi l'ultimo valore M[n,m] indica la similarità tra le due sequenze. La riga e la colonna Ø della M corrispondenti a prefissi vuoti di X e Y si inizializzano con valori negativi relativi alle sequenze di spazi ponendo:

$M[Ø,j]=-j$ per $Ø \le j \le m$, $M[i,Ø]=-i$ per $Ø \le i \le n$.

Indicato con p(i,j) il peso locale tra x_i e y_j, cioè posto $p(i,j)=+1$ se $x_i=y_j$ e $p(i,j)=-1$ se $x_i \neq y_j$, gli elementi M[i,j] si calcolano con la formula ricorsiva:

$$\mathbf{M[i,j] = max\{M[i,j-1]-1,\ M[i-1,j]-1,\ M[i-1,j-1]+p(i,j)\}} \qquad (2)$$

il cui significato dovrebbe essere ormai chiaro. Ovvia è la trasformazione della formula per pesi diversi da +1 e -1.

Utilizzando l'esempio precedente con X = ALBERO e Y = LABBRO otterremo la matrice:

		Ø	L	A	B	B	R	O
	Ø	Ø	-1	-2	-3	-4	-5	-6
A		-1	-1	Ø	-1	-2	-3	-4
L		-2	Ø	-1	-1	-2	-3	-4
B		-3	-1	-1	Ø	Ø	-1	-2
E		-4	-2	-2	-1	-1	-1	-2
R		-5	-3	-3	-2	-2	Ø	-1
O		-6	-4	-4	-3	-3	-1	+1

L'ultimo valore M[6,6] = +1 indica la similarita tra le due sequenze è uguale a +1.

La formulazione dell'algoritmo in forma di un programma è un'ovvia variazione del programma ED. Gli allineamenti che danno luogo alla similarità tra X e Y si ricostruiscono risalendo dalla cella M[n,m] alla M[Ø,Ø] come per la edit distance. Nell'esempio due allineamenti hanno similarità +1: li indichiamo con i costi puntuali associati:

```
A  L  -  B  E  R  O        -  A  L  B  E  R  O
-  L  A  B  B  R  O        L  A  B  B  -  R  O
-1 +1 -1 +1 -1 +1 +1      -1 +1 -1 +1 -1 +1 +1
```

Si noti che l'allineamento

```
A L B E R O
L A B B R O
```

che aveva la sessa edit distance dei primi due ha ora similarità 0 e non si ricava dalla matrice perché nella cella M[6,6] dà conto di allineamenti migliori.

Esercizio 3. Riempire manualmente la matrice M per determinare la similarità tra due sequenze a scelta di lunghezza almeno dieci costruite su un alfabeto di quattro caratteri, con pesi +1 match, -1 mismatch, -2 carattere contro spazio, frequentemente impiegati per sequenze di DNA.

Un altro problema rilevante in biologia molecolare, che impiega la stessa definizione di similarità del problema di global comparison e ne è in certo modo un'estensione, è noto come:

LOCAL COMPARISON. Date due sequenze X,Y trovare gli allineamenti di **massima similarità** tra una sottosequenza di X e una sottosequenza di Y.

Questo problema potrebbe sembrare molto più difficile del precedente perché non si conoscono a priori le sottosequenze di X e Y candidate alla soluzione, che diverranno note solo come risultato dell'algoritmo di confronto. Tuttavia il problema è risolubile con il consueto impiego della matrice M con qualche semplicissima variazione.

Dette S_X e S_y due sottosequenze di X e Y da confrontare per stabilirne la similarità, notiamo che dovranno essere confrontate a partire dal loro inizio tralasciando tutti i caratteri che le precedono in X e Y. Questo proprietà è stata già incontrata nella ricerca di un patttern in un testo in cui però solo i caratteri iniziali del testo andavano trascurati. Estendendo il criterio lì visto dovremo ora inizializzare con Ø tutti gli elementi nella prima riga e nella prima colonna di M, cioè porre:

$M[Ø,j]=Ø$, per $Ø \leq j \leq m$; $M[i,Ø]=Ø$, per $Ø \leq i \leq n$.

Nel caso presente il valore contenuto nella cella M[i,j] deve indicare la massima similarità tra due sottosequenze S_X, S_y che terminano in i,j, cioè $S_X = x_h \ldots x_i$ e $S_y = y_k \ldots y_j$ per opportuni valori $1 \leq h \leq i$, $1 \leq k \leq j$(ovvero S_X è un suffisso - o tratto finale - del prefisso $x_1\, x_2 \ldots x_i$ di X, e S_y è un suffisso del prefisso $y_1\, y_2 \ldots y_j$ di Y); e come sempre tali sottosequenze possono contenere degli spazi. L'osservazione chiave a questo punto è che se tutte le possibili sottosequenze S_X, S_y di almeno un carattere che terminano in i,j hanno similarità negativa, la similarità massima è zero perché si ha per S_X, S_y entrambe vuote. Il calcolo di M[i,j] si ottiene quindi con la formula (si noti lo zero nel calcolo del massimo):

$$\mathbf{M[i,j] = \max\{M[i,j-1]-1,\ M[i-1,j]-1,\ M[i-1,j-1]+p(i,j),\ 0\}} \quad (3)$$

Il calcolo in M si sviluppa come nei casi precedenti impiegando ora la formula (3). **Il risultato potrà ora apparire in qualsiasi cella della matrice**: si sceglieranno le celle M[i,j] contenenti i valori più alti, che indicano la terminazione in quel punto di due sottosequenze con tale similarità. La ricostruzione dell'allineamento, e quindi l'individuazione delle due sottosequenze "simili", avverrà con il solito metodo di tracciamento all'indietro arrestandosi quando si incontra una cella con valore zero; tuttavia in questo caso si possono ottenere anche altre informazioni come ora vedremo.

Consideriamo l'esempio X = ALTERO e Y = TALALTRA. Otterremo la matrice:

	Ø	T	A	L	A	L	T	R	A
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
A	Ø	Ø	+1	Ø	+1	Ø	Ø	Ø	+1
L	Ø	Ø	Ø	+2	+1	+2	+1	Ø	Ø
T	Ø	+1	Ø	+1	+1	+1	+3	+2	+1
E	Ø	Ø	Ø	Ø	Ø	Ø	+2	+2	+1
R	Ø	Ø	Ø	Ø	Ø	Ø	+1	+3	+2
O	Ø	Ø	Ø	Ø	Ø	Ø	Ø	+2	+1

I punti di massima similarità si incontrano nelle celle M[3,6]=+3 e M[5,7]=+3. Dalla prima si risale all'indietro lungo un solo percorso raggiungendo la cella M[0,3] che contiene zero, individuando così due sottosequenze identiche ALT in X e Y allineate senza spazi. Da M[5,7] si risale fino alla stessa cella M[0,3] in un solo percorso che corrisponde all'allineamento con similarità +3:

S_X = A L T E R
S_y = A L T - R

che comprende le due sottosequenze ALT già individuate in M[3,6]. Naturalmente il metodo individua anche coppie di sottosequenze con similarità minore della massima, come le due coppie di sottosequenze AL relative alle celle M[2,3] e M[2,5], allineate con similarità +2.

Esercizio 4. Ricostruire gli allineamenti tra sottosequenze relativi ai valori positivi di similarità nella matrice qui sopra.