Corso di Perfezionamento

La più lunga sottosequenza comune — LCS

Maria Rita Di Berardini[1]

[1]Dipartimento di Matematica e Informatica
Università di Camerino

17 febbraio 2009

Definizione del problema

Nelle applicazioni biologiche spesso si confronta il DNA di due, o più, organismi differenti

La struttura del DNA è formata da una stringa di molecole dette **basi**: adenina, citosina, guanina, e timina

La struttura del DNA è rappresentata da una stringa sull'insieme finito $\{A, C, G, T\}$

Il DNA di due organismi

$$S_1 = ACCGGTCGCGCGGAAGCCGGCCGAA$$

$$S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA$$

Definizione del problema

Uno degli scopi del confronto tra due molecole di DNA è quello di determinare il grado di somiglianza delle due molecole

Potremmo dire che due molecole si somigliano se, date le stringhe S_1 e S_2 che rappresentato le molecole:

- una è sottostringa di un'altra
- il numero delle modifiche richieste per trasformare l'una nell'altra è piccolo
- esiste una terza stringa S_3 le cui basi si trovano in ciascuna delle stringhe S_1 ed S_2: le basi devono presentarsi nello stesso ordine, senza essere necessariamente consecutive

Definizione del problema

Ad esempio, date le stringhe:

$$S_1 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA$$

$$S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA$$

la stringa S_3 è:

$$S_3 = GTCGTCGGAAGCCGGCCGAA$$

Formalizziamo questo concetto

Definizione del problema

Siano $X = \langle x_1, x_2, \cdots, x_m \rangle$ e $Z = \langle z_1, z_2, \cdots, z_k \rangle$ due sequenze; diciamo che Z è una **sottosequenza** di X se esiste una sequenza crescente $i_1, i_2, \cdots, i_k$ di indici di X tali che $x_{i_j} = z_j$ per ogni $j = 1, 2, \cdots, k$

Ad esempio, se $X = \langle A, B, C, B, D, A, B \rangle$ e $Z = \langle B, C, D, B \rangle$ la sequenza di indici è: $\langle 2, 3, 5, 7 \rangle$

$x_{i_1} = x_2 = B$, $x_{i_2} = x_3 = C$, $x_{i_3} = x_5 = D$ e $x_{i_4} = x_7 = B$

Definizione del problema

Date due sequenza X e Y diciamo che Z è una **sottosequenza comune** di X e Y se è una sottosequenza sia di X che di Y

Esempio: se $X = \langle A, B, C, B, D, A, B \rangle$ e $Y = \langle B, D, C, A, B, A \rangle$ la sequenza $\langle B, C, A \rangle$ è una sottosequenza di X eY, ma non è la più lunga (Longest Common Subsequence, LCS)

Sottosequenze comuni più lunghe: $\langle B, C, B, A \rangle$ e $\langle B, D, A, B \rangle$

Problema:
Date le sequenze $X = \langle x_1, x_2, \cdots, x_m \rangle$ ed $Y = \langle y_1, y_2, \cdots, y_n \rangle$, trovare una sottosequenza di lunghezza massima che è comune sia a X che a Y

Una prima soluzione: la forza bruta

In questo caso, significa considerare tutte le possibili sottosequenze di X e confrontarle con tutte le sottosequenze di Y

Quante sono le possibili sottosequenze di $X = \langle x_1, x_2, \cdots, x_m \rangle$?

Ogni sottosequenza di X corrisponde ad un sottoinsieme degli indici $\{1, 2, \cdots, m\}$

Tutti i possibili sottoinsiemi di $\{1, 2, \cdots, m\}$ sono 2^m. Quindi, **solo** enumerare tutte le possibili sottosequenze di X richiede un tempo esponenziale

Di nuovo, la tecnica della forza bruta non è adeguata

Fase 1: struttura della soluzione ottima

Dati $X = \langle x_1, x_2, \cdots, x_m \rangle$ e $i = 1, ..., m$, sia $X_i = \langle x_1, x_2, \cdots, x_i \rangle$ l'i-esimo prefisso di X (con $X_m = X$ e $X_0 = \langle\rangle$, la sequenza vuota)

Date $X = \langle x_1, x_2, \cdots, x_m \rangle$ ed $Y = \langle y_1, y_2, \cdots, y_n \rangle$, quale è la struttura di $Z = LCS(X, Y) = LCS(X_m, Y_n)$??

Caso banale $X = \langle\rangle$ o $Y = \langle\rangle$: allora, $Z = \langle\rangle$

Caso non banale $X, Y \neq \langle\rangle$: distinguiamo due casi:

1. $x_m = y_n$: accodiamo x_m alla sequenza $LCS(X_{m-1}, Y_{n-1})$
2. $x_m \neq y_n$: $Z = \max\{LCS(X, Y_{n-1}), LCS(X_{m-1}, Y)\}$

Fase 1: struttura della soluzione ottima

Dati $X = \langle x_1, x_2, \cdots, x_m \rangle$ e $i = 1, ..., m$, sia $X_i = \langle x_1, x_2, \cdots, x_i \rangle$ l'i-esimo prefisso di X (con $X_m = X$ e $X_0 = \langle\rangle$, la sequenza vuota)

Date $X = \langle x_1, x_2, \cdots, x_m \rangle$ ed $Y = \langle y_1, y_2, \cdots, y_n \rangle$, quale è la struttura di $Z = LCS(X, Y) = LCS(X_m, Y_n)$??

Caso banale $X = \langle\rangle$ o $Y = \langle\rangle$: allora, $Z = \langle\rangle$

Caso non banale $X, Y \neq \langle\rangle$: distinguiamo due casi:

1. $x_m = y_n$: accodiamo x_m alla sequenza $LCS(X_{m-1}, Y_{n-1})$
2. $x_m \neq y_n$: $Z = \max\{LCS(X, Y_{n-1}), LCS(X_{m-1}, Y)\}$

Sottostruttura ottima: Z contiene soluzioni ottime di sottoproblemi

L'idea:

La sottostruttuta ottima ci consente di risolvere il problema origina cercando LCS di coppie di prefissi di X e Y

Dobbiamo risolvere sottoproblemi della forma:

$$LCS(X_i, Y_j) \text{ con } i = 0 \cdots m \text{ e } j = 0 \cdots n$$

In totale sono $(n+1) \cdot (m+1) = O(nm)$ sottoproblemi distinti

Fase 2: una soluzione ricorsiva

Sia $c[i,j]$ la lunghezza di una qualsiasi LCS tra X_i e Y_j:

- se $i = 0$ o $j = 0$ (almeno una delle due è vuota) $c[i,j] = 0$
- se $i,j > 0$ e $x_i = y_j$, $c[i,j] = c[i-1,j-1] + 1$
- se $i,j > 0$ e $x_i \neq y_j$, $c[i,j] = \max\{c[i-1,j], c[i,j-1])\}$

Fase 2: una soluzione ricorsiva

Sia $c[i,j]$ la lunghezza di una qualsiasi LCS tra X_i e Y_j:

- se $i = 0$ o $j = 0$ (almeno una delle due è vuota) $c[i,j] = 0$
- se $i,j > 0$ e $x_i = y_j$, $c[i,j] = c[i-1,j-1] + 1$
- se $i,j > 0$ e $x_i \neq y_j$, $c[i,j] = \max\{c[i-1,j], c[i,j-1])\}$

$$c[i,j] = \begin{cases} 0 & \text{se } i = 0 \text{ o } j = 0 \\ c[i,j] = c[i-1,j-1] + 1 & \text{se } i,j > 0 \text{ e } x_i = y_j \\ \max\{c[i-1,j], c[i,j-1])\} & \text{se } i,j > 0 \text{ e } x_i \neq y_j \end{cases}$$

Fase 3: calcolare la lunghezza di una LCS

Esempio: calcolare la lunghezza di una LCS di $X = \langle A, B, C, B, D, A, B \rangle$ e $Y = \langle B, D, C, A, B, A \rangle$

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0						
B	2	0						
C	3	0						
B	4	0						
D	5	0						
A	6	0						
B	7	0						

Casi Base: $c[i, 0] = c[0, j] = 0$ per ogni $i = 1, ..., m$ e $j = 1, ..., n$

Fase 3: calcolare la lunghezza di una LCS

Se $i, j > 0$ di quali posizioni abbiamo bisogno per calcolare $c[i, j]$

- se $x_i = y_j$, abbiamo bisogno di $c[i-1, j-1]$ (posizione in alto e a sinistra)
- se $x_i \neq y_j$. abbiamo bisogno di $c[i-1, j]$ (in alto) e $c[i, j-1]$ (a sinistra)

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0						
B	2	0						
C	3	0		$\nwarrow$	$\uparrow$			
B	4	0		$\leftarrow$	i, j			
D	5	0						
A	6	0						
B	7	0						

Ci basta riempire la matrice riga per riga

Fase 3: calcolare la lunghezza di una LCS

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0						

Fase 3: calcolare la lunghezza di una LCS

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0	0	0	0	1	1	1
B	2	0						

Fase 3: calcolare la lunghezza di una LCS

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0	0	0	0	1	1	1
B	2	0	1	1	1	1	2	2

Fase 3: calcolare la lunghezza di una LCS

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0	0	0	0	1	1	1
B	2	0	1	1	1	1	2	2
C	3	0	1	1	2	2	2	2
B	4	0	1	1	2	2	3	3
D	5	0	1	2	2	2	3	3
A	6	0	1	2	2	3	3	4
B	7	0	1	2	2	3	4	4

La lunghezza di una LCS tra X e Y è data da $c[n, m]$

L'algoritmo per il calcolo della lunghezza di una LCS

LcsLength(X, Y)

1. $m \leftarrow length[X]$
2. $n \leftarrow length[Y]$
3. **for** $i \leftarrow 1$ **to** m **do** $c[i, 0] \leftarrow 0$
4. **for** $j \leftarrow 1$ **to** n **do** $c[0, j] \leftarrow 0$
5. **for** $i \leftarrow 1$ **to** m
6. **do for** $j \leftarrow 1$ **to** n
7. **do if** $x_i = y_j$
8. **then** $c[i, j] \leftarrow c[i-1, j-1] + 1$
9. **else if** $c[i-1, j] \geq c[i, j-1]$
10. **then** $c[i, j] \leftarrow c[i-1, j]$
11. **else** $c[i, j] \leftarrow c[i, j-1]$
12. **return** c

Il calcolo di ogni posizione della tabella richiede un tempo costante; allora il costo di esecuzione della procedura è $O(nm)$

Fase 4: costruzione di una LCS

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0	0	0	0	1	1	1
B	2	0	1	1	1	1	2	2
C	3	0	1	1	2	2	2	2
B	4	0	1	1	2	2	3	3
D	5	0	1	2	2	2	3	3
A	6	0	1	2	2	3	3	4
B	7	0	1	2	2	3	4	4

Possiamo riesaminare le scelte fatte per calcolare la lunghezza di una LCS al contrario, ossia dall'ultima alla prima, per costruire una soluzione ottima

Fase 4: costruzione di una LCS

			B	D	C	A	B	A
		0	1	2	3	4	5	6
	0		0	0	0	0	0	0
A	1	0	0	0	0	1	1	1
B	2	0	↖ **1**	← 1	1	1	2	2
C	3	0	1	1	↖ **2**	← 2	2	2
B	4	0	1	1	2	2	↖ **3**	3
D	5	0	1	2	2	2	↑ 3	3
A	6	0	1	2	2	3	3	↖ **4**
B	7	0	1	2	2	3	4	↑ 4

L'algoritmo per il calcolo della lunghezza di una LCS

LcsLength(X, Y)

1. $m \leftarrow length[X]$
2. $n \leftarrow length[Y]$
3. **for** $i \leftarrow 1$ **to** m **do** $c[i, 0] \leftarrow 0$
4. **for** $j \leftarrow 1$ **to** n **do** $c[0, j] \leftarrow 0$
5. **for** $i \leftarrow 1$ **to** m
6. **do for** $j \leftarrow 1$ **to** n
7. **do if** $x_i = y_j$
8. **then** $c[i, j] \leftarrow c[i-1, j-1] + 1$
9. $b[i, j] \leftarrow$ " $\nwarrow$ "
10. **else if** $c[i-1, j] \geq c[i, j-1]$
11. **then** $c[i, j] \leftarrow c[i-1, j]$
12. $b[i, j] \leftarrow$ " $\uparrow$ "
13. **else** $c[i, j] \leftarrow c[i, j-1]$
14. $b[i, j] \leftarrow$ " $\leftarrow$ "
15. **return** b e c

L'algoritmo per la costruzione di una LCS

PrintLCS(b, X, i, j)

1. **if** $i = 0$ or $j = 0$ **then return**
2. **if** $b[i, j] =$ "$\nwarrow$"
3. **then PrintLCS**$(b, X, i-1, j-1)$
4. stampa x_i
5. **else if** $b[i, j] =$ "$\uparrow$"
6. **then PrintLCS**$(b, X, i-1, j)$
7. **else PrintLCS**$(b, X, i, j-1)$

La chiamata iniziale è **PrintLCS**$(b, X, length[X], length[Y])$

La procedura impiega $O(n+m)$, poichè almeno una delle due dimensioni coinvolte diminuisce ad ogni passo della ricorsione