Progettazione di Algoritmi **Anno Accademico 2021–2022**

Note per la Lezione 17

Ugo Vaccaro

In questa lezione studieremo il problema della Più Corta Supersequenza di due sequenze. Date due sequenze $a = a[1]...a[m]$ e $b = b[1]...b[n]$ di caratteri, il problema è di trovare la lunghezza della più corta supersequenza (PCS) che contiene a e b come sottosequenze. Detto in altri termini, cerchiamo una sequenza $c = c[1]...c[k]$, con k *minimo*, tale che, per opportuni interi $1 \leq i_1 < i_2 < \ldots i_m \leq k$ e $1 \leq j_1 < j_2 < \ldots j_n \leq k$ valga che

$$c[i_1] = a[1], c[i_2] = a[2], \ldots, c[i_m] = a[m] \quad \text{e} \quad c[j_1] = b[1], c[j_2] = b[2], \ldots, c[j_n] = b[n]$$

Ad esempio, se $a = ABCBDAB$ e $b = BDCABA$, allora la lunghezza della PCS di a e b è 9. Possibili PCS sono $ABCBDCABA$, $ABDCABDAB$ e $ABDCBDABA$.

Infatti

$$\begin{aligned} a &= ABCBDAB \\ \mathsf{PCS}(a,b) &= ABCBDCABA \\ b &= BDCABA \end{aligned}$$

Per sequenze arbitrarie x e y, denotamo con $\mathsf{PCS}(x, y)$ una generica più corta supersequenza (PCS) che contiene x e y come sottosequenze e con $|\mathsf{PCS}(x, y)|$ la sua lunghezza (che vogliamo trovare).

Per risolvere il problema mediante la Programmazione Dinamica, deriviamo una formulazione ricorsiva della sua soluzione. Condideriamo due casi.

Caso 1: $a[m] = b[n]$.
In questo caso, possiamo far vedere che

$$\mathsf{PCS}(a[1]...a[m], b[1]...b[n]) = \mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1]) + a[m],$$

dove + denota l'operazione di concatenazione.

Per provare ciò, iniziamo con il denotare con α una $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$, ed osserviamo che α deve necessariamente terminare con $a[m]$, altrimenti non sarebbe una più corta supersequenza comune ad $a[1]...a[m]$ e $b[1]...b[n]$.

Quindi sappiamo che α è della forma $\alpha = \beta + a[m]$. Osserviamo ora che β non è una sequenza arbitraria, ma è pari ad una $\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])$, ovvero è una soluzione ottima al sottoproblema corrispondente alle due sequenze $a[1]...a[m-1]$ e $b[1]...b[n-1]$

Che β sia una supersequenza comune ad $a[1]...a[m-1]$ e $b[1]...b[n-1]$ è ovvio, visto che tutti i caratteri di $a[1]...a[m]$ e $b[1]...b[n]$ appaiono in $\alpha = \beta + a[m] = \beta + b[n]$, e quindi tutti i caratteri di $a[1]...a[m-1]$ e $b[1]...b[n-1]$ appaiono in β.

Di conseguenza

$$|\beta| \geq |\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])|,$$

visto che $|\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])|$ è la lunghezza della più corta supersequenza comune a $a[1]...a[m-1]$ e $b[1]...b[n-1]$.

Se β non fosse una $\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])$ (ovvero se non avesse lunghezza minima tra tutte le supersequenza comuni a $a[1]...a[m-1]$ e $b[1]...b[n-1]$, allora dovrebbe valere che

$$|\beta| > |\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])|.$$

Detto in altri termini, esisterebbe un'altra supersequenza comune a $a[1]...a[m-1]$ e $b[1]...b[n-1]$, sia essa γ tale che

$$|\beta| > |\gamma| = |\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])|.$$

Ne vien fuori che $\gamma + b[n]$ è una supersequenza comune a $a[1]...a[m]$ e $b[1]...b[n]$ di lunghezza

$$|\gamma| + 1 < |\beta| + 1 = |\alpha|$$

contro l'ipotesi che α è una $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$.

Ad esempio $\mathsf{PCS}(ABCBDA, BDCABA) = \mathsf{PCS}(ABCBD, BDCAB) + A$

$\mathsf{PCS}(ABCBDAB, BDCAB) = \mathsf{PCS}(ABCBDA, BDCA) + B$.

Consideriamo ora il secondo caso.

Caso 2. $a[m]$ è differente da $b[n]$.

Partiamo dalla semplice osservazione che una generica $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$, detta essa α, o termina con $a[m]$ o termina con $b[n]$. Infatti sia $a[m]$ che $b[n]$ devono apparire in una qualsiasi $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$ (altrimenti essa non sarebbe una supersequenza comune a $a[1]...a[m]$ e $b[1]...b[n]$) ed almeno uno tra $a[m]$ che $b[n]$ deve apparire nell'ultima posizione di α (altrimenti α non sarebbe la più corta).

Possiamo allora concludere che una $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$ o è una $\mathsf{PCS}(a[1]...a[m], b[1]...b[n-1])$ con alla fine la concatenzazione del simbolo $b[n]$ oppure è una $\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n])$ con alla fine la concatenzazione del simbolo $a[m]$.

Visto che abbiamo appreso che una generica $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$ o termina con $a[m]$ o con $b[n]$, assumiamo che essa termini con $a[m]$ (il caso in cui termina con $b[n]$ è perfettamente analogo).

Sia una tale più corta supersequenza pari ad $\alpha = \beta + a[m]$. Ovviamente β è una supersequenza di $a[1]...a[m-1]$ e $b[1]...b[n]$. Quello che vogliamo provare è equivalente a dire che β è una $\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n])$. Se ciò non fosse, ovvero se $|\beta| > |\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n])|$, allora avremmo che una generica $\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n])$, con $a[m]$ alla fine, sarebbe una supersequenza comune a $a[1]...a[m]$ ed a $b[1]...b[n]$ di lunghezza inferiore ad α, contro l'ipotesi che α è una $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$.

Ovviamente noi non sappiamo se una $\mathsf{PCS}(a[1]...a[m], b[1]...b[n])$ termina con $a[m]$ o con $b[n]$. Risolviamo il problema nel solito modo: ci calcoliamo $|\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n]) + a[m]|$ e $|\mathsf{PCS}(a[1]...a[m], b[1]...b[n-1]) + b[n]|$ e ci prendiamo il minimo di tali due valori.

Mettendo tutto insieme, e ricordando che vogliamo calcolare $|\mathsf{PCS}(a[1]...a[m], b[1]...b[n])|$, otteniamo che

$$|\mathsf{PCS}(a[1]...a[m], b[1]...b[n])| = \begin{cases} |\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n-1])| + 1 & \text{se } a[n] = b[m] \\ \min\{|\mathsf{PCS}(a[1]...a[m-1], b[1]...b[n])| + 1, & \\ \qquad |\mathsf{PCS}(a[1]...a[m], b[1]...b[n-1])| + 1\} & \text{se } a[n] \neq b[m] \end{cases}$$

con il caso base $|\mathsf{PCS}(a[1]...a[m], b[1]...b[n])| = m + n, \quad$ se $\; m = 0 \quad$ oppure $\; n = 0$.

L'algoritmo ricorsivo è ovvio.

```
Calcola_PCS(a[1...i], b[1...j])      % usa una tabella pcs[,]
IF(i==0||j==0) {
RETURN i+j
    } ELSE {
       IF(pcs[i,j] non è definito) {
         IF(a[i]==b[j]) {
           pcs[i,j]=pcs[i-1,j-1]+1
          } ELSE {
           pcs[i,j]=min{pcs[i-1,j]+1, pcs[i,j-1]+1}
        }
      }
   }
RETURN pcs[i,j]
```

La complessità dell'algoritmo è chiaramente $\Theta(nm)$. Per esercizio, si progetti ed analizzi un algoritmo che prendendo in input sequenze a e b e la tabella `p` calcolata dall'algoritmo `Calcola_PCS`$(a[1...n], b[1...m])$, produca in output una $\mathsf{PCS}(a[1...n], b[1...m])$.

◇

Date due sequenze $s = s[1] \dots s[n]$ e $p = p[1] \dots p[m]$, diremo che p è una sottosequenza di s se esistono indici $1 \leq i_1 < i_2 < \dots < i_m \leq n$ tali che $s[i_1] = p[1], s[i_2] = p[2], \dots, s[i_m] = p[m]$.

Il problema che vogliamo studiare è quello di calcolare il numero di volte che p compare come sottosequenza di s.

Ad esempio, per $s =$*subsequence* e $p =$*sue*, come output vorremmo 7, in quanto:
1. subsequence
2. subsequence
3. subsequence
4. subsequence
5. subsequence
6. subsequence
7. subsequence

Formuliamo il problema in termini ricorsivi.

Se confrontiamo l'ultimo carattere di $s = s[1]...s[n]$ con l'ultimo carattere di $p = p[1]...p[m]$, ci sono due possibilità.

- Se l'ultimo carattere di s è *uguale* all'ultimo carattere di p, allora andiamo a contare il numero di volte in cui $p[1]...p[m-1]$ appare in $s[1]...s[n-1]$ (ognuno di questi ci darà una distinta occorrenza di $p = p[1]...p[m]$ in $s = s[1]...s[n]$). A questo numero occorre sommare il numero di volte in cui $p = p[1]...p[m]$ occorre in $s[1]...s[n-1]$ (perchè sono occorrenze differenti da quelle prima calcolate, che si riferivano solo alle occorrenze di p in s in cui i due ultimi simboli apparivano entrambi nell'ultima posizione).
- Se l'ultimo carattere di s è *diverso* dall'ultimo carattere di p, allora andiamo a contare il numero di volte in cui $p = p[1]...p[m]$ occorre in $s[1]...s[n-1]$

Denotiamo con $c[i,j]$ il il numero di volte che $p[1]...p[j]$ compare come sottosequenza di $s[1]...[i]$. La relazione di ricorrenza sarà:

$$c[i,j] = \begin{cases} c[i-1,j-1] + c[i-1,j] & \text{se } a[i] = p[j] \\ c[i-1,j] & \text{se } a[i] \neq p[j] \end{cases}$$

I casi base sono i seguenti:

1. $c[0,0] = 1$
2. $c[i,0] = 1, \forall i > 0$
3. $c[0,j] = 0, \forall j > 0$

Il codice dell'algoritmo di Programmazione Dinamica e la sua analisi è per esercizio.