Tecniche Algoritmiche: programmazione dinamica

Una breve presentazione

Programmazione dinamica

Tecnica bottom-up:

1. Identifica dei sottoproblemi del problema originario, procedendo logicamente dai problemi più piccoli verso quelli più grandi
2. Utilizza una tabella per memorizzare le soluzioni dei sottoproblemi incontrati: quando si incontra lo stesso sottoproblema, sarà sufficiente esaminare la tabella
3. Si usa quando i sottoproblemi non sono indipendenti, e lo stesso sottoproblema può apparire più volte

Esempio: numeri di Fibonacci

Un “case study”

Problema: calcolo della istanza tra stringhe

Siano X e Y due stringhe di lunghezza m ed n:

$$X = x_1 \cdot x_2 \cdot \cdots \cdot x_m \qquad Y = y_1 \cdot y_2 \cdot \cdots \cdot y_n$$

Vogliamo calcolare la “distanza” tra X e Y, ovvero il minimo numero delle seguenti operazioni elementari che permetta di trasformare X in Y

`inserisci`(a):	Inserisci il carattere a nella posizione corrente della stringa.
`cancella`(a):	Cancella il carattere a dalla posizione corrente della stringa.
`sostituisci`(a, b):	Sostituisci il carattere a con il carattere b nella posizione corrente della stringa.

Esempio

Azione	Costo	Stringa ottenuta
Inserisco P	1	P|*RISOTTO*
Mantengo R	0	PR|*ISOTTO*
Sostituisco I con E	1	PRE|*SOTTO*
Mantengo S	0	PRES|*OTTO*
Cancello O	1	PRES|*TTO*
Mantengo T	0	PREST|*TO*
Cancello T	1	PREST|*O*
Mantengo O	0	PRESTO|

Approccio

- Denotiamo con $\delta(X,Y)$ la distanza tra X e Y
- Definiamo X_i il prefisso di X fino all'i-esimo carattere, per $0 \leq i \leq m$ (X_0 denota la stringa vuota):

$$X_i = x_1 \cdot x_2 \cdot \cdots \cdot x_i \text{ se } i \geq 1$$

- Risolveremo il problema di calcolare $\delta(X,Y)$ calcolando $\delta(X_i,Y_j)$ per ogni i, j tali che $0 \leq i \leq m$ e $0 \leq j \leq n$
- Manterremo le informazioni in una tabella D di dimensione $m \times n$

Inizializzazione della tabella

- Alcuni sottoproblemi sono molto semplici
- $\delta(X_0,Y_j)=j$ partendo dalla stringa vuota X_0, basta inserire uno ad uno i j caratteri di Y_j
- $\delta(X_i,Y_0)=i$ partendo da X_i, basta rimuovere uno ad uno gli i caratteri per ottenere Y_0
- Queste soluzioni sono memorizzate rispettivamente nella prima riga e nella prima colonna della tabella D

Avanzamento nella tabella (1/3)

- Se $x_i = y_j$, il minimo costo per trasformare X_i in Y_j è uguale al minimo costo per trasformare X_{i-1} in Y_{i-1}

$$D[i,j] = D[i-1, j-1]$$

- Se $x_i \neq y_j$, distinguiamo in base all'ultima operazione usata per trasformare X_i in Y_j in una sequenza ottima di operazioni

Avanzamento nella tabella (2/3)

$\texttt{inserisci}(y_j)$: il minimo costo per trasformare X_i in Y_j è uguale al minimo costo per trasformare X_i in Y_{j-1} più 1 per inserire il carattere y_j

⇒ $D[i,j] = 1 + D[i, j-1]$

$\texttt{cancella}(x_i)$: il minimo costo per trasformare X_i in Y_j è uguale al minimo costo per trasformare X_{i-1} in Y_j più 1 per la cancellazione del carattere x_i

⇒ $D[i,j] = 1 + D[i-1, j]$

Avanzamento nella tabella (3/3)

$\texttt{sostituisci}(x_i, y_j)$: il minimo costo per trasformare X_i in Y_j è uguale al minimo costo per trasformare X_{i-1} in Y_{j-1} più 1 per sostituire il carattere x_i per y_j

⇒ D [i,j] = 1 + D [i-1 , j-1]

In conclusione:

$$D[i,j] = \begin{cases} D[i-1, j-1] & \text{se } x_i = y_j \\ 1 + \min\{D[i, j-1], D[i-1, j], D[i-1, j-1]\} & \text{se } x_i \neq y_j \end{cases}$$

Pseudocodice

algoritmo `distanzaStringhe`$(stringa\ X, stringa\ Y) \rightarrow intero$

matrice D di $(m+1) \times (n+1)$ interi
for $i = 0$ **to** m **do** $D[i,0] \leftarrow i$
for $j = 1$ **to** n **do** $D[0,j] \leftarrow j$
for $i = 1$ **to** m **do**
  for $j = 1$ **to** n **do**
    if $(x_i \neq y_j)$ **then**
      $D[i,j] \leftarrow 1 + \min\{D[i,j-1], D[i-1,j], D[i-1,j-1]\}$
    else $D[i,j] \leftarrow D[i-1,j-1]$
return $D[m,n]$

Tempo di esecuzione ed occupazione di memoria: O(m n)

Esempio di esecuzione

		P	**R**	**E**	**S**	**T**	**O**
	0	**1**	2	3	4	5	6
R	1	1	**1**	2	3	4	5
I	2	2	2	**2**	3	4	5
S	3	3	3	3	**2**	3	4
O	4	4	4	4	**3**	3	3
T	5	5	5	5	**4**	**3**	4
T	6	6	6	6	5	**4**	4
O	7	7	7	7	6	5	**4**

Consideriamo la tabella D costruita dall'algoritmo.

In grassetto sono indicate due sequenze di operazioni che permettono di ottenere la distanza tra le stringhe

Conclusione

Programmazione dinamica: varie applicazioni: calcolo dei numeri di Fibonacci, distanza tra stringhe, associatività del prodotto tra matrici, cammini minimi tra tutte le coppie di nodi (algoritmo di Floyd e Warshall)