Lezione 30 aprile

Thursday, April 30, 2020 9:00 AM

Programmazione Dinamica

LCS

$$LCS(i,j) = \begin{cases} LCS(i-1, j-1) + 1 & \text{se } a_i = b_j \\ \max(LCS(i, j-1), LCS(i-1, j)) \end{cases}$$

Principio di ottimalità

Teorema:

$X = x_1 \dots x_n$

$Y = y_1 \dots y_m$

$Z = z_1, \dots, z_k$ $\quad LCS(X, Y)$

1) se $x_n = y_m$ allora $z_k = x_n = y_m$

e $z_1, \dots, z_{k-1}$ è $LCS(x_1 \dots x_{n-1}, y_1 \dots y_{m-1})$

2) se $x_n \neq y_m$

allora $z_k \neq x_n$ implica

$z_1, \dots, z_k$ è $LCS(x_1 \dots x_{n-1}, y_1 \dots y_m)$

3) se $x_n \neq y_m$

allora $z_k \neq y_m$ implica

$z_1, \dots, z_k$ è $LCS(x_1 \dots x_n, y_1 \dots y_{m-1})$

Prova

1) se $z_k \neq x_n$

$z_1 \dots z_k \, x_n$

$|W| \geq k$

2) $z_k \neq x_m$

Z è LCS $(x_1 \dots x_{n-1}, y_1 \dots y_m)$

essendo

W di lunghezza $> k$

è anche LCS $(x_1 \dots x_n, y_1 \dots y_m)$

$X : x_1, \dots, x_n$

$Y : y_1, \dots, y_m$ $\quad n \simeq m$

EDIT DISTANCE

ALLINEAMENTO OTTIMO

col numero minore di errori

1) MISMATCH $x_i \neq y_j$ 1

2) inserzione : $x_i \perp$ 1

3) cancellaz. $\perp y_j$ 1

1) $ED(i,j) = ED[x_1 \dots x_i, y_1 \dots y_j]$

$ED(n,m) = ED[X,Y]$

2) $ED(0,j) = j \quad 0 \le j \le m$

$ED(i,0) = i \quad 0 \le i \le n$

$Y_1 \dots Y_j$

$X \quad \sqcup \dots \sqcup$

$X =$ ALBERO

$Y =$ LABBRO

x x = x = =

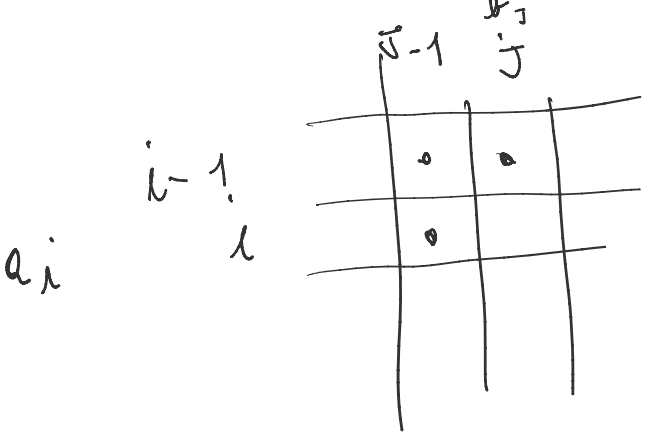
distanza = 3

X ALBERO

Y LABBRO

x = x = x = =

distanza = 3



$$p_{ij} = \begin{cases} 0 & \text{se } a_i = b_j \\ 1 & \text{se } a_i \neq b_j \end{cases}$$

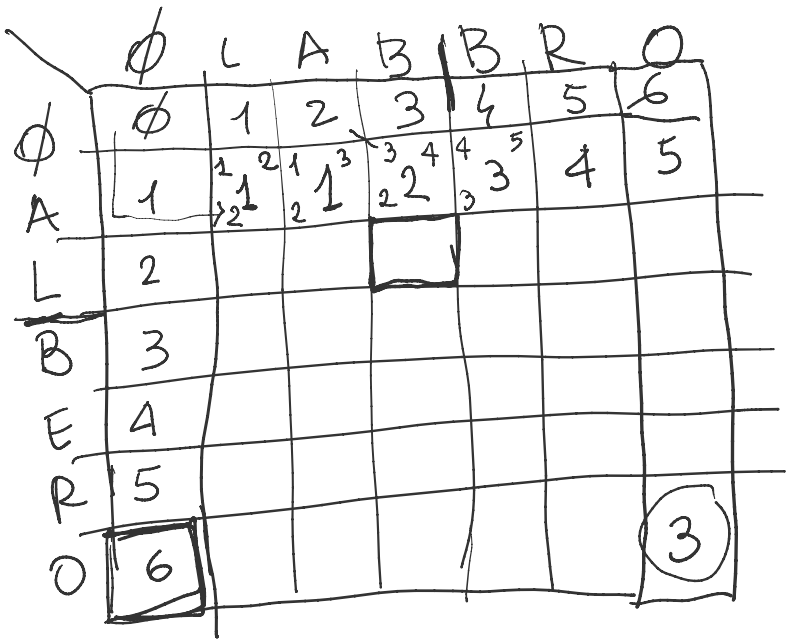
$$ED[i,j] = \min \begin{cases} ED[i-1,j-1] + p_{ij} \\ ED[i,j-1] + 1 \\ ED[i-1,j] + 1 \end{cases}$$

4) costruisci tabella

$E[(n+1) \times (m+1)]$

$Y =$ LA⊔BBRO

⊔



ALBERO

distanza 6

	A ␣	␣ A
A	A ␣	␣ A
L	␣ L	L ␣
1	2	2

$\Theta(n \cdot m)$

Spazio $\Theta(n, m)$

se vogliamo l'allineamento

Spazio $\Theta(n)$

- completare esempio
- scrivere l'alg. che calcola ED
- scrivere l'alg. che produce l'allineamento
- ottimalità

se in input sono presenti blank e allineo un blank di input con un blank assegnato ⇒ errore = 0;

0-1 Knapsack

Zaino, bisaccia, ladro

input: $S = s_1 \dots s_n$
$v_1 \dots v_n$ valore (in €)
$p_1 \dots p_n$
Z = peso max trasportabile

output: determinare un sottoinsieme $S' \subseteq S$:

$$\sum_{s_i \in S'} v_i \text{ sia max}$$

$$\sum_{s_i \in S'} p_i \leq Z$$

0-1 Knapsack è difficile (NP-hard)

$Z = 8$

	peso	valore	val/peso
oggetto 1	5	10	2
oggetto 2	4	6	1.5
oggetto 3	4	6	1.2

oggetto 3 4 ∨ 1

Algoritmi greedy

Selezione l'oggetto a valore/peso valore max finché c'è posto nello zaino

Soluzione { oggetto 1 } valore 10

dà la soluzione ottima per Zaino frazionabile

0-1 Knapsack → Progr. Dinamica
n elementi v_i, p_j, W

$1 \le i \le n, 1 \le j \le Z$

1) $Z(i,j)$ = il valore massimo di una soluzione che considera un sottoinsieme degli oggetti $1..i$, e che ha un peso $\le j$.

$Z(n,W)$ = soluzione al problema 0-1 Knapsack.

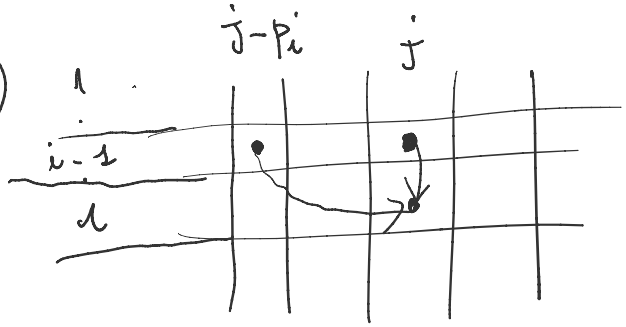
2) $Z(i,0) = \emptyset$

$Z(0,j) = \emptyset$

4) $Z(0..n, 0..W)$ per la memorizzazione delle sottosoluzioni

delle sottosoluzioni

3)



2

$$Z(i,j) = \begin{cases} \text{non riesce nella sol. } s_i \\ \text{riesce} \end{cases} \qquad \max \begin{cases} Z(i-1, j) \\ Z(i-1, j-p_i) + v_i \end{cases}$$

$s_i = (v_i, p_i)$ — riesce s_i — > 0

ES

$W = 5$

	a_1	a_2	a_3
v:	60	100	120
p:	1	2	3

a \ W	0	1	2	3	4	5
0	0	0	0	0	0	0
a_1	0	60	60	60	60	60
a_2	0	60	100	160	160	160
a_3	0	60	100	160	180	220

$\{a_3, a_2\}$

$$\max \begin{cases} Z(i-1, j) = Z(1, 2) = 60 \\ Z(i-1, j-p_i) + v_i = 100 \end{cases} = 100$$

$\max(60, 160) = 160$

$\max(160, 120) = 160$

$\max(160, 180) = 180$

$\max(160, 180) = 180$

$\max(160, 220) = 220$ Vel. sol. ottima