Progettazione di algoritmi

Programmazione Dinamica

ESERCIZIO 1

Data una matrice binaria di dimensioni $n \times n$ vogliamo verificare se nella matrice è possibile raggiungere la cella in basso a destra partendo da quella in alto a sinistra senza mai toccare celle che contengono il numero 1.

Si tenga conto che dalla generica cella (i, j) ci si può spostare solo nella eventuale cella in basso (vale a dire la cella $(i+1, j)$) o nella eventuale cella a destra (vale a dire la cella $(i, j+1)$).

Progettare un algoritmo che risolve il problema in tempo $O(n^2)$.

Ad esempio: per la matrice A la risposta deve essere SI mentre per la matrice B la risposta deve essere NO.

$$A = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 \\ \hline \end{array} \qquad B = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

ESERCIZIO 2

Data una matrice di dimensione $n \times n$ le cui celle sono numerate con numeri distinti che vanno da 1 a n^2, vogliamo trovare la massima lunghezza possibile per cammini che toccano celle con numerazione crescente e incremento di 1.

I cammini possono partire da una qualunque cella e, nel corso del cammino, dalla generica cella (i, j) ci si può spostare in una qualunque cella adiacente in orizzontale o verticale (vale a dire in una delle celle $(i, j+1), (i+1, j), (i, j-1), (i-1, j)$). La lunghezza di un cammino è data dal numero di nodi toccati dal cammino.

Progettare un algoritmo che risolve il problema in tempo $O(n^2)$

Ad esempio: per la matrice A la risposta è 1 mentre per la matrice B, grazie al cammino $2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$, la risposta è 6

$$A = \begin{array}{|c|c|c|} \hline 3 & 6 & 2 \\ \hline 7 & 1 & 9 \\ \hline 4 & 8 & 5 \\ \hline \end{array} \qquad B = \begin{array}{|c|c|c|} \hline 9 & 7 & 6 \\ \hline 8 & 2 & 5 \\ \hline 1 & 3 & 4 \\ \hline \end{array}$$

ESERCIZIO 3

Il problema dello zaino: Abbiamo uno zaino di capacità C ed n oggetti, ognuno con un peso p_i e un valore v_i. Vogliamo trovare un sottoinsieme degli oggetti la cui somma dei pesi non supera C e che massimizza il valore totale.

Esempio: Considera la seguente istanza con $C = 11$ e $n = 5$ oggetti con peso e valore riportati in tabella

oggetto	valore	peso
1	1	1
2	6	4
3	18	5
4	22	5
5	28	7

- il sottoinsieme di oggetti $\{3, 5\}$ non è una soluzione ammissibile (perché di peso $12 > C$)

- il sottoinsieme di oggetti $\{1, 2, 4\}$ è una soluzione ammissibile (di peso 10 e valore 29) ma non ottima (perché ne esistono di valore maggiore).

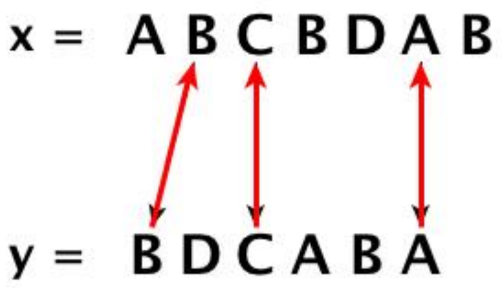
- il sottoinsieme di oggetti $\{1, 3, 4\}$ è una soluzione ammissibile (di peso 11 e valore 41) si può dimostrare che è anche una soluzione ottima.

ESERCIZIO 4

Il problema della massima sottosequenza comune. Date due sequenze di simboli X e Y, vogliamo trovare una sottosequenza comune di lunghezza massima.

ESEMPIO per $X = ABCBDAB$ e $Y = BDCABA$

- una delle possibili sottosequenze comuni di lunghezza 3:



- ci sono due diverse sottosequenze comuni di lunghezza massima B, C, A, B e B, C, B, A: