Ottimizza la somma

Supponete di avere in input un vettore di n interi positivi distinti $V[1 \dots n]$ e un valore W. Scrivere un algoritmo che:

1. restituisca il massimo valore $X = \sum_{i=1}^{n} x[i]V[i]$ tale che $X \leq W$ e ogni $x[i]$ è un intero non negativo;
2. stampi il vettore x.

Ad esempio, per $V[\,] = \{18, 3, 21, 9, 12, 24\}$ e $W = 17$, una possibile soluzione ottima è $x = [0, 2, 0, 1, 0, 0]$ da cui deriva $X = 15$.
Discutere correttezza e complessità.

Mosse su scacchiera

- Supponete di avere una scacchiera di dimensione $n \times n$ che rappresenta una scacchiera e un pedone che dovete muovere dall'estremità inferiore a quella superiore.
- Il pedone si può muovere (1) una casella in alto, oppure (2) una casella in diagonale alto-destra, oppure (3) una casella in diagonale alto-sinistra.
- Alla scacchiera è associata una matrice di profitto P; quando una cella (r, c) viene visitata, si guadagna un profitto reale $P[r][c]$

Scrivere un algoritmo che restituisca il massimo profitto ottenibile partendo da una qualunque cella dell'estremità inferiore e raggiungendo una qualunque cella dell'estremità superiore, seguendo le regole appena descritte.

	1	2	3	4	5
1	6	**7**	4	7	$\underline{8}$
2	7	**6**	1	1	$\underline{4}$
3	3	5	**7**	$\underline{8}$	2
4	2	**6**	$\underline{7}$	0	2
5	**7**	3	5	$\underline{6}$	1

I Promessi Sposi

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte; e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni."

Quante volte questo testo contiene la sottosequenza "lucia"?

I Promessi Sposi

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte; e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni."

Quante volte questo testo contiene la sottosequenza "lucia"?
Alcune considerazioni:

- Due sottosequenze sono distinte (e quindi vanno contate separatamente) se esiste almeno una differenza negli insiemi di caratteri utilizzati.
- Esempio: "did you go" contiene due volte la sottosequenza "dog"....

I Promessi Sposi

Scrivere un algoritmo che prenda in input una stringa testo T di n caratteri e una stringa pattern P di m caratteri e restituisca il numero di volte distinte che la stringa P appare come sottosequenza di T.

Discutere correttezza e complessità dell'algoritmo.

Spoiler alert!

Ottimizza la somma

E' possibile notare che questo problema è un caso particolare dello Zaino senza limiti di scelta, quindi – a livello di compito – è possibile semplicemente chiamare il codice che abbiamo discusso a lezione, passando il vettore D sia come peso che come profitto.

---

int bestSum(**int**[ ] V, **int** n, **int** W)

---

return knapsack(V, V, n, W)

La complessità è $O(nW)$.

Ottimizza la somma

Risolviamo invece il problema "da capo". Sia $DP[i][w]$ il massimo valore che posso ottenere seguendo le regole di cui sopra avendo a disposizione i primi i oggetti e un valore massimo w.

$$DP[i][w] = \begin{cases} -\infty & i \geq 0 \wedge w < 0 \\ 0 & i = 0 \vee w = 0 \\ \max\{DP[i-1][w], DP[i][w - V[i]] + V[i]\} & \text{altrimenti} \end{cases}$$

La complessità (spaziale e temporale) della soluzione è $O(nW)$. Si può adattare un approccio simile a quello visto per lo zaino senza limiti per ridurre la complessità spaziole a $O(W)$.

Ottimizza la somma

int bestSum(**int**[ ] V, **int** n, **int** W)

% Crea un vettore DP inizializzato a -1
int[ ][ ] DP = **new int**$[0 \ldots n][1 \ldots W] = \{-1\}$
return bsRec(V, n, W, DP)

int bsRec(**int**[ ] V, **int** i, **int** w, **int**[ ][ ] DP)

if $w < 0$ **then**
  return $-\infty$
if $i == 0$ **or** $w == 0$ **then**
  return 0
if $DP[i][w] < 0$ **then**
  $DP[i][w] = \max(\text{bsRec}(V, i-1, w, DP), \text{bsRec}(V, i, w - V[i], DP) + V[i])$
return $DP[i][w]$

Mosse su scacchiera (Soluzione 2.7 in 13-pd.pdf)

Definiamo una matrice DP tale che $DP[r][c]$ rappresenta il massimo guadagno che si può ottenere partendo da una cella (r, c) e arrivando ad una cella della riga in alto ($r = 1$), dove r e c rappresentano la riga e la colonna, rispettivamente.

Il valore di ritorno corrisponderà al valore massimo dell'ultima riga ($r = n$).

Per esercizio, scrivere un algoritmo che restituisca anche il percorso, e non solo il valore massimo.

$$DP[r][c] = \begin{cases} -\infty & c < 1 \textbf{ or } c > n \\ P[1][c] & r = 1 \textbf{ and } 1 \leq c \leq n \\ \max \begin{pmatrix} DP[r-1][c-1], \\ DP[r-1][c], \\ DP[r-1][c+1] \end{pmatrix} + P[r][c] & 2 \leq r < n \textbf{ and } 1 \leq c \leq n \end{cases}$$

La complessità dell'algoritmo risultante, che deve riempire ogni cella della tabella, é $\Theta(n^2)$.

Mosse su scacchiera

```
int searchPath(int[][] P, int n)
int DP = new int[1...n][1...n]
for c = 1 to n do                          % Copia la prima riga nel vettore DP
    DP[1][c] = P[1][c]
for r = 2 to n do
    for c = 1 to n do
        DP[r][c] = -∞
        foreach d ∈ {-1, 0, +1} do
            int newc = c + d
            if 1 ≤ newc ≤ n then
                DP[r][c] = max(DP[r][c], DP[r - 1][newc] + P[r][c])
return max(DP[n])                          % Restituisce il massimo dell'ultima riga
```

I Promessi Sposi

Sia $DP[i][j]$ il numero di occorrenze del prefisso j-esimo del pattern $P(j)$ come sottosequenza del prefisso i-esimo del testo $T(i)$.

$$DP[i][j] = \begin{cases} 0 & i = 0 \textbf{ and } j > 0 \\ 1 & j = 0 \\ DP[i-1][j] + DP[i-1][j-1] & i > 0 \textbf{ and } j > 0 \textbf{ and } T[i] = P[j] \\ DP[i-1][j] & i > 0 \textbf{ and } j > 0 \textbf{ and } T[i] \neq P[j] \end{cases}$$

- Se il testo è finito ($i = 0$) e il pattern non è vuoto ($j > 0$), non siamo riusciti a trovare il pattern
- Se il pattern è vuoto ($j = 0$), significa che siamo riusciti a trovare il pattern, e lo contiamo per uno
- Se l'ultimo carattere del testo e del pattern sono uguali, possiamo sfruttare quest'uguaglianza oppure no; i due casi vanno sommati
- Se l'ultimo carattere del testo e del pattern sono diversi, ignoriamo l'ultimo carattere del testo.

I Promessi Sposi

Utilizziamo un vettore $DP[0 \dots m]$ invece che una matrice, in quanto il valore si ottiene a partire dalla sola riga precedente, che viene memorizzata in DP'.
Complessità in tempo: $\Theta(mn)$, complessità in spazio $\Theta(m)$

---

int lucia(ITEM[ ] T, ITEM[ ] P, **int** n, **int** m)

```
int[] DP = new int[0...m]
int[] DP' = new int[0...m]
DP[0] = 1
for j = 1 to m do DP[j] = 0
for i = 1 to n do
    for j = 0 to m do DP'[j] = DP[j]
    for j = 1 to m do
        if T[i] == P[j] then
            DP[j] = DP'[j] + DP'[j − 1]
        else
            DP[j] = DP'[j]
return DP[m]
```

---

5 Maggio e Promessi Sposi

Subject: Sottosequenza Promessi sposi
Date: Fri, 11 Dec 2015 00:25:41 +0100
To: Alberto Montresor <alberto.montresor@unitn.it>

Il 5 maggio nei promessi sposi senza considerare spazi, punteggiatura e numeri, ma considerando gli accenti ci sta:

219754653015166309795736175938257695138575835630252623797897783379476158171917574339842832197562154239662334744219763715818422816009859675872567801017800136565973566045203119340799723612777962220975263078675519750712637479432237655210391918486018747374239424385310182137281795662107004225375841957767155366649493437946947434130448636772119986920548451717813640098831758107771539361489284456030355662857753722957658724970416074137670807561854959314707635812057348445330682675118608161304322179728660590437840811285379588850669382000672869551575023563015330128593082577269020619288952011873970086359263850877345042074027243309950696549344467341096985080369133555862845505929949281872847365683962633684668376711431054269109960860130141804038350182348913395557865326952783427223436474143160403851682665482045722458282856692545688317000656065623166181081105288235575975572352074726528752376937507087958987383644703249684014961465095879761604854835120500024685074595216118769351618590697862390987987264814086004487458833092023307

Allego lo script in python che ha generato il risultato (con testo completo dei promessi sposi e del 5 maggio). Ovviamente ha tempi assurdi per svariati motivi.