Mosse su scacchiera

Supponete di avere una scacchiera $n \times n$ e un pedone che dovete muovere dall'estremità inferiore a quella superiore. Un 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. Non può tornare indietro. Quando una cella (x, y) viene visitata, guadagnate un valore reale $p(x, y)$.
Calcolare un percorso da una qualunque casella dell'estremità inferiore ad una qualunque casella dell'estremità superiore, massimizzando il profitto.

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

Palindroma

Una stringa si dice palindroma se è uguale alla sua trasposta, cioè se è identica se letta da sinistra e destra o da destra a sinistra.
Scopo dell'esercizio è scrivere una funzione $f(s)$ che ritorna il numero minimo di caratteri da **inserire** in s necessari per rendere s palindroma.
Per esempio, input: "casacca":

- $n = 7$ caratteri: "casaccaACCASAC"
- $n = 6$ caratteri: "casaccaCCASAC"
- $n = 3$ caratteri: "casaccaSAC"
- $n = 2$ caratteri: "ACcasacca"

Notate che non necessariamente i caratteri si inseriscono in testa o in fondo; per esempio, "anta" $\rightarrow$ "antNa".
Fornire un algoritmo per calcolare $f()$. Discutere la complessità dell'algoritmo.

Costo partizione di un vettore

Costo $C(i,j)$ di un sottovettore $V[i \dots j]$ di V $C(i,j) = \sum_{t=i}^{j} V[t]$
k-partizione di V è una divisione di V in k sottovettori
$V[1,j_1], V[j_1+1,j_2], V[j_2+1,j_3], \dots, V[j_{k-1}+1,j_k]$ con $j_k = n$ e
$j_t < j_{t+1}, \forall 1 \leq t < k$, ovvero tale per cui i sottovettori coprono totalmente il vettore e non si sovrappongono.
Costo della k-partizione è il costo massimo dei suoi sottovettori.
Dato un vettore V di n interi e un intero k,con $2 \leq k \leq n$, trovare una k-partizione di V di costo minimo
Esempio: $V = \{2,3,7,-7,15,2\}$, $k = 3$
1: $\{2,3,7\}, \{-7,15\}, \{2\}$, costo $2+3+7 = 12$
2: $\{2,3\}, \{7\}\{-7,15,2\}$, costo $-7+15+2 = 10$

1. Soluzione per $k = 2$ (suggerimento: $O(n)$).
2. Soluzione per $k = 3$ (suggerimento: $O(n^2)$).
3. Soluzione generale (suggerimento: $O(kn^2)$).

Spoiler alert!

Mosse su scacchiera

```
search-path(integer[][] p, integer n)
integer DP = new integer[1...n][1...n]
for x = 1 to n do
    DP[x][n] = p[x][n]
for y = n − 1 downto 1 do
    for x = 1 to n do
        DP[x, y] = −∞
        foreach d ∈ {−1, 0, +1} do
            integer x' = x + d
            if x' ≥ 1 and x' ≤ n then
                integer t = DP[x'][y + 1] + p[x, y]
                if t > DP[x][y] then
                    DP[x][y] = t
return max({DP[x][1]|1 ≤ x ≤ n|})
```

Palindroma

- Se la stringa $s = \mathrm{a}s'\mathrm{a}$ è composta da due identici caratteri "a" iniziale e finale, allora:

$$f(s) = f(s')$$

- Se la stringa $s = \mathrm{a}s'\mathrm{b}$ ha due caratteri iniziale e finale diversi, aggiungiamo o un carattere "b" in testa (e consideriamo il problema $\mathrm{a}s'$, oppure aggiungiamo un carattere "a" in coda (e consideriamo il problema $s'\mathrm{b}$). Scegliamo fra le due possibilità quella con costo minore. In entrambi i casi, dobbiamo sommare 1 per il carattere aggiunto.

$$f(s) = \min\{f(\mathrm{a}s'), f(s'\mathrm{b})\} + 1$$

Palindroma

```
calcolaF(ITEM[] s, integer[][] F, integer i, integer j)
if j ≤ i then
    return 0
else if F[i, j] = ⊥ then
    if s[i] = s[j] then
        F[i, j] ← calcolaF(s, i + 1, j − 1)
    else
        F[i, j] ← min(calcolaF(s, i, j − 1), calcolaF(s, i + 1, j)) + 1
    return F[i, j]
```

2-partizione

2-partition(**integer**[ ] V, **integer** n)

integer $tot \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  $tot \leftarrow tot + V[i]$
integer $sofar \leftarrow 0$
integer $min \leftarrow +\infty$
for $i \leftarrow 1$ **to** $n-1$ **do**
  $sofar \leftarrow sofar + V[i]$
  $min \leftarrow \mathsf{min}(min, \mathsf{max}(sofar, tot - sofar))$
return min

3-partizione

```
integer 3-partition(integer[] V, integer n)
integer[][] T ← new integer[0...n]
T[0] ← 0
for i ← 1 to n do
    T[i] ← T[i − 1] + V[i]
integer min ← +∞
for i ← 1 to n − 2 do
    for j ← i + 1 to n − 1 do
        integer temp ← max(T[i], T[j] − T[i], T[n] − T[j])
        min ← min(min, temp)
return min
```

k-partizione

Sia $M[i,t]$ il minimo costo associato al sottoproblema di trovare la migliore t-partizione nel vettore $V[1 \ldots i]$. Il problema iniziale corrisponde a $M[n,k]$ – ovvero trovare la migliore k-partizione in $V[1 \ldots n]$. Sfruttiamo un vettore di appoggio T definito come nel caso $k = 3$.

$$M[i,t] = \begin{cases} T[i] & t = 1 \\ +\infty & t > i \\ \min_{1 \le j < i} \max(M[j, t-1], T[i] - T[j]) & \text{altrimenti} \end{cases}$$

k-partizione

integer partition-rec(**integer**[ ] V, **integer**[ ] T, **integer**[ ][ ] M, **integer** i, **integer** t)

```
if t > i then
    return +∞
if t = 1 then
    return T[i]
if M[i, t] = ⊥ then
    integer M[i, t] ← +∞
    for j ← 1 to i − 1 do
        integer temp ← max(partition-rec(V, T, M, j, t − 1), T[i] − T[j])
        if temp < M[i, t] then
            M[i, t] ← temp
return M[i, t]
```